SPARX SYSTEMS

ENTERPRISE ARCHITECT

User Guide Series

Guide to Decision Modeling

Author: Sparx Systems & Stephen Maguire
Date: 2026-05-28
Version: 17.1

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Guide to Decision Modeling 5
Introduction to Decision Modeling 7
Getting Started 18
Setting Up a Model Structure 22
Tailoring the Application for Decision Modeling 24
Creating Diagrams Elements and Relationships 31
Windows and Tools 39
Decision Model and Notation Overview 46
What is Decision Model and Notation 48
A First Example 51
Levels of Usage 55
Why Use Decision Model and Notation 58
When to Use Decision Model and Notation 60
An Example Decision Model 67
Benefits of Decision Model and Notation 70
Visualize Decisions and Rules 72
Facilitate Collaboration 74
Simplify Business Process Diagrams 76
Communicate Shared Understanding 78
Discover Opportunities for Automation 80
Incorporate into Architecture Models 82
Create a Single Source of Truth for Decisions 84
Simulate Decision Models 86
Test Decision Models 87

Generate Programming Code 88
Check Consistency and Correctness 89
Find Gaps in Specifications 91
Articulate with other Models 94
Visualize Inputs and Analytics 95
Synopsis of the Notation 96
The Decision Requirements Diagram 103
Decision Expression Types 115
Decision Table 118
Literal Expression 121
Boxed Context 123
Invocation 125
Expression Languages 128
Expression Editor 129
Friendly Enough Expression Language (FEEL) 132
Decision Tables Explained 138
Table Orientation 141
Allowable Value Fields 143
Data Types for Input Output Clauses 145
Rules and Inputs and Outputs 146
Hit Policies 148
Merging and Unmerging Cells 155
Validating a Decision Model 157
Gaps in Rules 160
Overlapping Rules 164
Context for Decision Model and Notation 166
Simulating a Decision Model 178

Setting up a Simulation 180
Running a Simulation 185
Simulating a Decision Service 192
Code Generation from a Decision Model 194

Guide to Decision Modeling

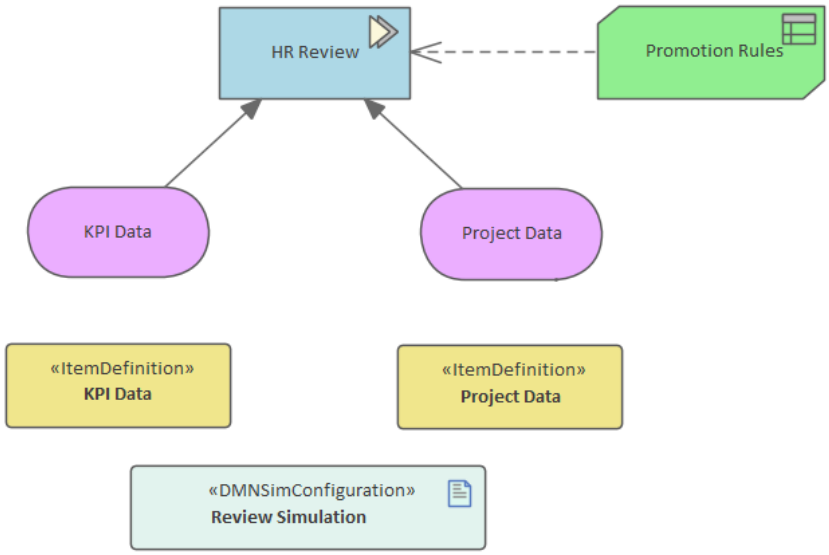
Every now and then a new standard is released that is destined to fundamentally change both the way we work and the quality of the business and technology systems that underpin almost every aspect of our lives. The *Decision Model and Notation* is such a standard and its implementation in the world's leading Enterprise Modeling tool paves the way for a renaissance in the delivery of software-centric enterprise, business, technology, engineering and scientific systems, large and small.

The standard helps remove the age-old communication crevasse between business and technology staff, and allows the business, at a strategic or tactical level, to define and manage the decisions that shape the outcomes of the organization and its customers and suppliers. It is a tool for all seasons and disciplines, starting from a strategist defining a question and the set of answers they want the system to operate with, down to a web service designer who simply presses a button to implement the decision. The value proposition is that the decisions can be modeled with diagrams and simple tables of rules with straightforward expressions such as 'Is a Customer's Monthly Disposable Income > $2000'. Once the rules have been defined they can be tested by the business using their own data without any involvement of the technical community. If a business person can create a simple spreadsheet they will find DMN extremely easy to use and will be able to create and validate their own decision models. The communication features

built into the platform facilitate immediate discussions with other business and technical stakeholders.

Introduction to Decision Modeling

Now more than ever, in a world turned on its head by new and innovative business and technical ideas and disruptive ways of working, does an organization need to have a clear understanding of its choices and the decisions it makes. Unmanaged complexity is the enemy and its opposite - *agility* - the friend that heralds business success and enables an organization to respond quickly to changes in its business circumstances. Without a clear and communicable model it is almost impossible for an organization to embrace the changes that confront it daily in the digital world. The description and implementation of decisions, which has been inexplicably and somewhat invisibly part of almost all other disciplines has now been synthesized into a rigorous and formal discipline of its own, with a new way of modeling and describing decisions, Inputs, Outcomes, Rules, Business Knowledge, Authorities and more. Indeed, once you have seen the Decision Model and Notation in action and been introduced to the countless benefits it brings, you will not be able to go back to old and arcane ways of working.



Enterprise Architect has become the tool of choice for many Business and Technical leaders because of its flexible, extensible, standards-based and pragmatic approach to modeling complex systems. As a collaboration platform it is a tool for all disciplines, and allows Decision Models to be created, integrated, managed, documented, simulated and generated to programming code. The models can be visualized and integrated with a range of other models including Business Process Diagrams, Use Case Models, User Stories, Test Cases, Database Models, Implementation Artifacts and programming code to list just the main models.

How it will help you

Most readers will typically have some knowledge of decision-making in an organization, but each reader will most likely have different experiences and a different way of defining, managing and working with decisions. Decisions arise across the whole fabric of organizational descriptions and implementation. Distilling the decisions

into a separate but articulated model will provide great clarity and value. The reader will benefit by understanding Enterprise Architect's features and the tools that are available to develop and manage Decision Models, which in turn will enable them to be more productive as an individual and as a member of a team. The ultimate value, though, will be to the organizations they work for, which will acquire the ability to respond to change efficiently and with agility, allowing them to navigate through the complex and ever-changing business environment of the digital age.

Who will benefit

Anyone involved in the development, management or implementation of decisions, whether at a strategic level, a business value level or a technology level, will benefit from reading this guidebook. This encompasses a wide range of roles, whose work and decisions will ultimately be guided and facilitated by the models, but specifically there are four groups for whom using Decision Model and Notation holds great value:

- **Strategic Thinkers,** who are responsible for steering the enterprise through the turbulent waters of change, setting the goals and understanding the drivers, will benefit by being able to visualize the decisions they make and know that they are being implemented according to their intentions.
- **Business and Process Analysts,** who currently have business rules and decisions built into their Process

diagrams in the form of complex and cascading sets of Gateways, will benefit because they will be able to extract these rules, placing them into an articulated Decision Model. The result will be a reduction of complexity, straight-through Process diagrams that are easier to understand and more resilient to change, and a simple model of their decisions that can be simulated, tested and implemented.

- **Software Engineers,** who know full well the benefit of isolating the business rules from the main body of the code and allowing them to be configured, will benefit because they not only will have a clear and tested model of the rules but they will also be able to automatically generate programming code that enshrines the rules, removing the possibility of errors of interpretation or translation.
- **System Engineers**, who are accustomed to complex and often seemingly intractable problems in a wide range of disciplines and contexts from space exploration to oceanography. Systems are designed to operate within a context that require decisions to be made, such as production line robots, planetary rovers, transport control systems, safety control systems in machinery plants and many more. These systems typically rely heavily on decisions to maintain operational efficiency, safety and to be able to respond to variables that change within the environments.

What you will learn

This topic will teach you how to use the many features of Enterprise Architect to develop and manage Decision Models using the new Decision Model and Notation (DMN), to connect the Decision Models to other types of enterprise model such as Business Process, StateMachine, Use Case and Parametric, to simulate them, to automatically generate programming code, to create documentation and to work collaboratively as a member of a team.

You will learn what tools are available, which tools should be used for a particular technique, and how to use them. For example, the topic will teach you how to decompose complex and seemingly intractable decision rules into a simple and understandable model, using diagrams and Decision Tables. These can be simulated and implemented manually or used to generate high-quality programming code in a wide range of languages, using the tools and facilities available within Enterprise Architect.

Overview of the Guide

This Guide is divided into a number of topics that will introduce you to Decision Modeling from a number of perspectives, ensuring that once you have worked through the document you will have a sound knowledge of the why, what and how of Decision Modeling. The Guide commences with a *Getting Started* topic that will introduce the concepts with broad brushstrokes, followed by an overview. Next is a list of benefits and a series of topics that develop the detail. The concepts, notation and tool usage are

all introduced, giving you the theoretical and practical knowledge to get started with your own Decision Models and to derive the benefit of applying this approach to modeling decisions.

Getting Started The *Getting Started* topic provides just enough information for you to begin setting up your own models, starting with setting up a model structure, tailoring the application , creating your first diagrams and learning how to work with the windows and tools used in modeling decisions in Enterprise Architect.

Decision Model and Notation Overview The *Decision Model and Notation Overview* topic introduces the DMN standard and provides a simple example. The topic continues by introducing the concept of levels of usage, which will help you see how it will work practically in an organization. The topic continues to provide the context for its usage, and when and why it should be used.

Benefits of Decision Model and Notation The *Benefits of Decision Model and Notation* topic is almost a business case for the use of this approach and exemplifies why an organization should use Enterprise Architect to model decisions. After completing the topic you

should be able to see how and why it could be used to assist your own organization in modeling decisions more formally and rigorously, and the benefits that would be gained from taking this approach, including validation and the generation of implementation code.

Context for Decision Model and Notation

The *Context for Decision Model and Notation* topic will help Modelers, Requirements Analysts and other stakeholders understand the situations in which the discipline of Decision Modeling can be used. It introduces a broad range of contexts from business, engineering and scientific disciplines. It discusses some of the canonical examples, such as the use of Decision Models with Business Process diagrams, and explores other interesting applications that are support by Enterprise Architect.

Example Decision Model

The *Example Decision Model* topic introduces a complete example that is simple enough to understand but also has sufficient complexity to demonstrate some of the expressive aspects of the standard and the facilities available in Enterprise Architect. The example will

provide a useful backdrop for some of the later topics that will introduce more of the richer features of the language and the tool.

Synopsis of the Notation

The *Synopsis of the Notation* topic introduces you to the Decision Model and Notation standard. This includes the visual elements that will be placed on diagrams, including elements, relationships and artifacts, and their meaning and usage.

The Decision Requirements Diagram

The *Decision Requirements Diagram* topic introduces the main diagram used for constructing Decision Models, and teaches you how to create, modify and work with the elements using the many tools available within Enterprise Architect. Many of the things you will learn in this topic can be applied to other diagram types as well, so by the final topic you will be well on your way to knowing how to work with the tool.

Decision Expression Types

The *Decision Expression Types* topic is where we explore how the logic for decisions is defined. It introduces the Expression types that can be used to describe the logic, and the expression

editor that can be used to manage these expressions.

Decision Tables Explained

The *Decision Tables Explained* topic drills down into the details of the commonest expression type - the Decision Table - and explains Hit Policies, allowable values, value and expression types and more. This will become an important reference for both novice and experienced Decision Modelers.

Validating a Decision Model

The *Validating a Decision Model* topic introduces the validation tool, which can be used to check the consistency, correctness and completeness of the model. This is a useful feature built-in to Enterprise Architect, providing a safeguard against errors due to gaps and overlaps in expressions. Validation should be used as a precursor to simulation to ensure that the models are sound, expressive and logically codify the intent and reasoning behind the business decisions.

Simulating a Decision Model

The *Simulating a Decision Model* topic introduces features that allows a Decision Model to be executed as though it were in

situ in a production system. Enterprise Architect allows a modeler (business or technical) to run the simulation with no need for configuration. Any decision within the Requirements diagram can be selected for simulation, including the highest level decision. You can select any predefined input data sets and run the simulation multiple times to see the model outputs with different data as inputs.

Code Generation from a Decision Model

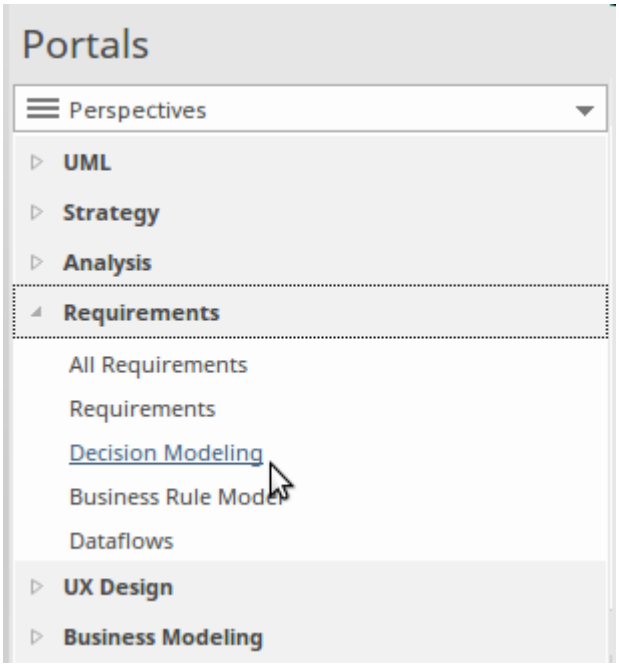
The *Code Generation from a Decision Model* topic introduces a productivity tool within Enterprise Architect that allows implementation (programming) code to be automatically generated directly from the model. The facility is key to the successful implementation of the rules in a runtime engine, as there is no need for programmers or technical staff to interpret the models or attend meetings with business users - the code is generated directly from the models. This facility will remove problems associated with the misinterpretation of business intent, that have plagued the industry. The workflow is straightforward - specify the decisions, define the expressions, validate the expressions and generate the

code.

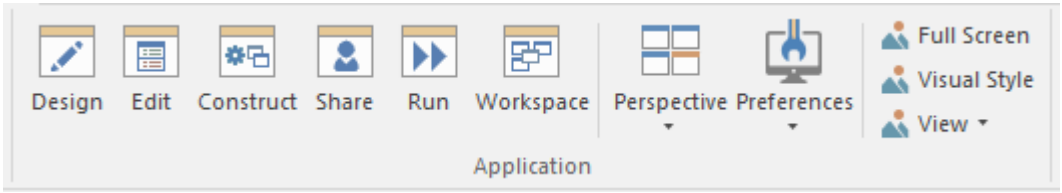
Getting Started

Getting started with a new tool is often one of the most difficult challenges facing a Business Analyst or Technical Analyst, but Enterprise Architect makes this easy by providing a number of facilities to assist the newcomer to the tool. Enterprise Architect is a large and multi-featured application and the breadth and depth of its coverage could overwhelm a person new to the program, but fortunately a solution to this has been built into the design.

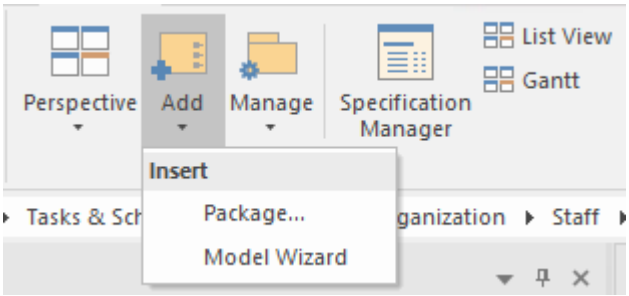
Perspectives can be used to limit the functionality to Decision Modeling - making it easy for a Requirements Analyst or other stakeholder to get started. You still have the ability to utilize other functionality that might be useful, such as Strategic Modeling, Requirements, Business Process Modeling, Mind Mapping and more, simply by changing Perspectives - all without having to open a different tool. It is worth noting that Perspectives exist for a wide range of modeling disciplines that Enterprise Architect supports. For more information see the Model Perspectives Help topic.



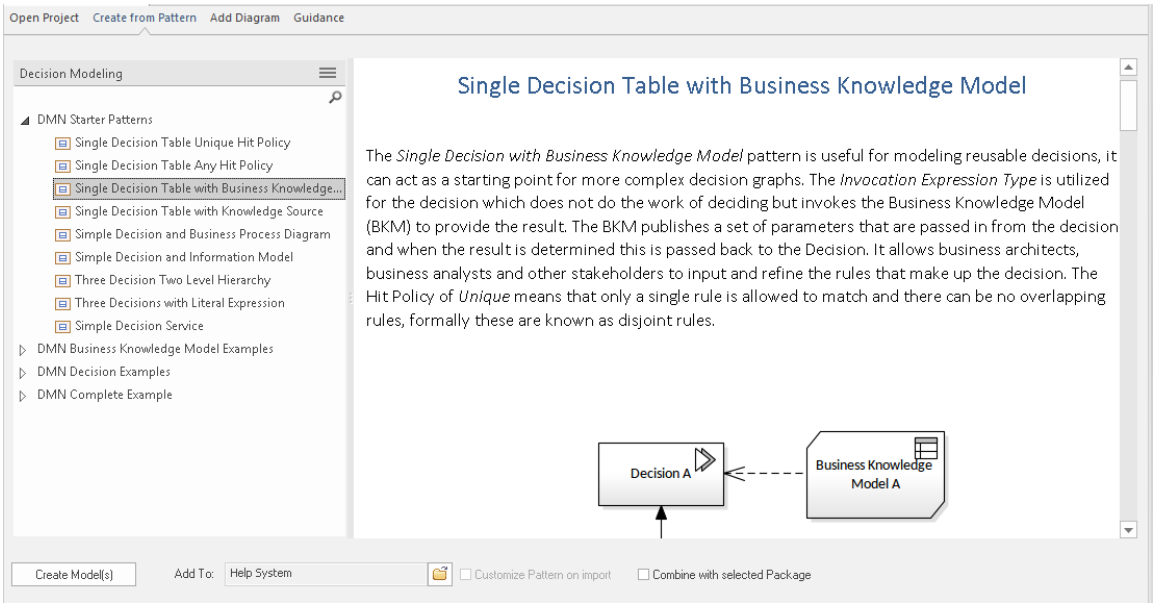
You also have tremendous flexibility in tailoring your own environment and the user interface by setting preferences and selecting workspaces and visual styles. For more information see the Advanced Customization Help topic.



Setting up a new project is straightforward with the use of the Model Builder patterns (with accompanying documentation) that can be used to automatically create a DMN project structure to get you started. The Model Builder can then be used to create any number of Decision Requirements diagrams as the model is developed and the problem and solution spaces are fleshed-out.



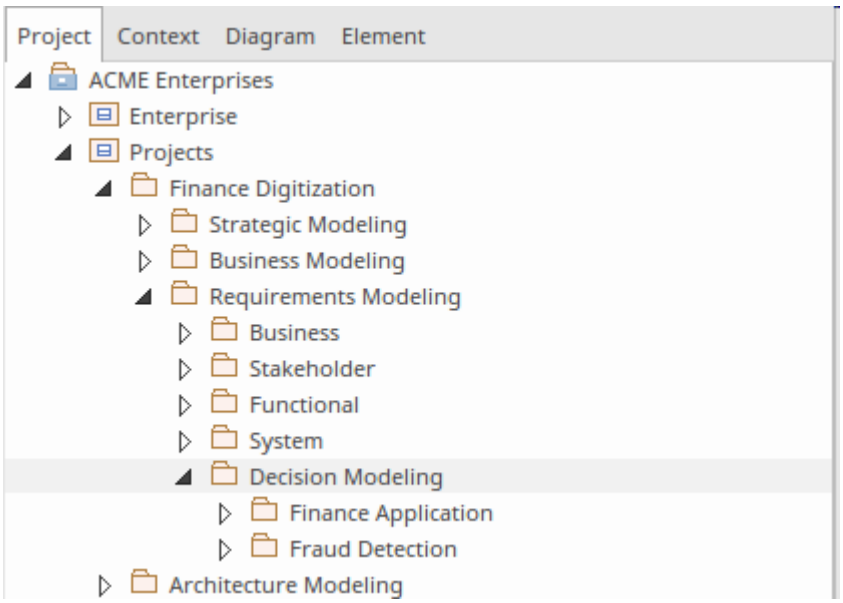
All of these facilities make it easy for a newcomer to get started, allowing you to become a productive member of a team and to start contributing to models quickly and without any delay. If you are a novice Requirements Analyst, you will be surprised how productive you can be when compared to working in text-based or other more rudimentary modeling tools. There will be challenges along the way as you push yourself and the tool to new limits, but a handy Help system, a large community of users, comprehensive forums, a community site and first class support services will make the journey easy and informative. This image shows one of the built-in model patterns for the *Decision Modeling* perspective, with a range of useful patterns that can be automatically injected into your own models. Each pattern comes with accompanying documentation that explains how to use the pattern and helps to ensure you follow best practice.


Open Project
Create from Pattern
Add Diagram
Guidance
Decision Modeling
DMN Starter Patterns
Single Decision Table Unique Hit Policy
Single Decision Table Any Hit Policy
Single Decision Table with Business Knowledge...
Single Decision Table with Knowledge Source
Simple Decision and Business Process Diagram
Simple Decision and Information Model
Three Decision Two Level Hierarchy
Three Decisions with Literal Expression
Simple Decision Service
DMN Business Knowledge Model Examples
DMN Decision Examples
DMN Complete Example
Single Decision Table with Business Knowledge Model
The Single Decision with Business Knowledge Model pattern is useful for modeling reusable decisions, it can act as a starting point for more complex decision graphs. The Invocation Expression Type is utilized for the decision which does not do the work of deciding but invokes the Business Knowledge Model (BKM) to provide the result. The BKM publishes a set of parameters that are passed in from the decision and when the result is determined this is passed back to the Decision. It allows business architects, business analysts and other stakeholders to input and refine the rules that make up the decision. The Hit Policy of Unique means that only a single rule is allowed to match and there can be no overlapping rules, formally these are known as disjoint rules.
Decision A
Business Knowledge Model A
Create Model(s)
Add To:
Help System
Customize Pattern on import
Combine with selected Package

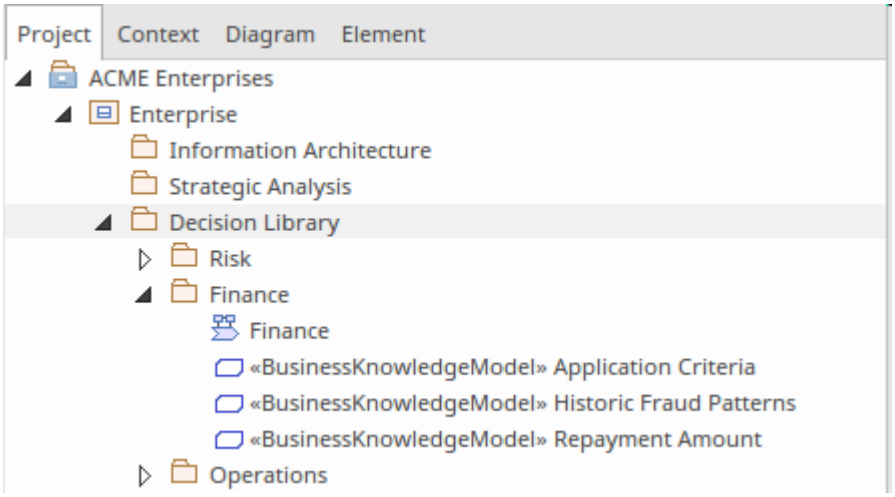
Setting Up a Model Structure

Enterprise Architect has been designed as a productivity tool from the ground up, and includes a built-in Model Builder providing a library of model patterns, that makes setting up a model structure very simple. These intuitive tools save valuable time for the experienced user and remove difficulties for the beginner. With Decision Modeling, the elements that you create will typically form part of a project or initiative and be related to other models, such as Business Process diagrams and StateMachines created by Business Analysts, System Engineers, Software Engineers and others.

This illustration shows an example repository structure that indicates the location of the Decision Modeling elements within the Requirements Modeling branch of a project. The Decision Models can be located anywhere within a project, and Enterprise Architect provides a facility to drag-and-drop them (or their containing Package) from one location to another within the Browser window.



The next illustration shows how Business Knowledge Models, which are designed to be reusable decision components, can be contained in a library and can be reused by any project or initiative. The library can be stored at the level of the enterprise models, and when needed the appropriate Business Knowledge Models can be included in the respective project by dragging them from the Browser window onto a Decision Requirements diagram (DRD) in the selected project.



The Business Knowledge Models provide a mechanism for reuse by allowing decision information to be passed in as parameters and the output to be passed back to the invoking decision.

The overall structure of the repository is a subject that is beyond the scope of this guidebook, but it is a critical aspect of setting up a repository and your librarian or systems administrator will have performed this task or will be able to assist you. We will learn later that Packages are important units in the organization and maintenance of a model repository. For more information see the The Model Builder Help topic.

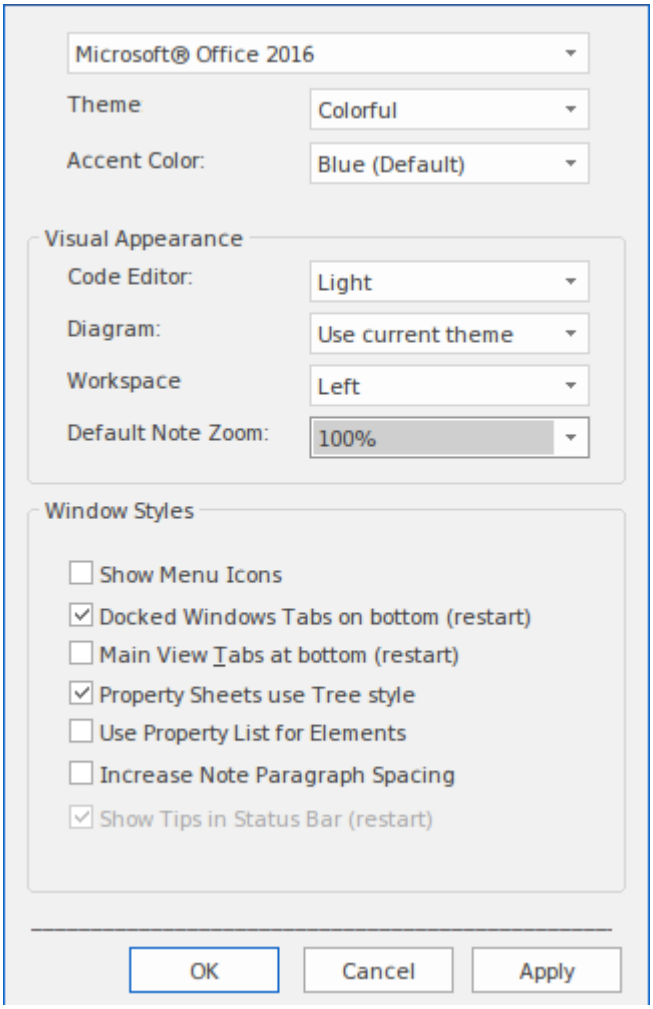
Tailoring the Application for Decision Modeling

Enterprise Architect is a tool with a vast amount of functionality, which is one of the reasons why it is so popular as a tool for modeling systems of any kind. To ensure that the tool provides the most benefit to an organization, team, project or individual, some tailoring of the interface to suit the modeling intent will ensure that all parties achieve the best outcomes. Most of the settings can be changed by a single button click, transforming the tool to be fit for purpose, which in our case is collaborating on projects where decisions need to be modeled, articulated and simulated.

In these sections we will look at a number of places where we can change the application from a generic modeling tool to a requirements modeling tool with a focus specifically on the creation, management and simulation of Decision Models.

Selecting a Visual Style

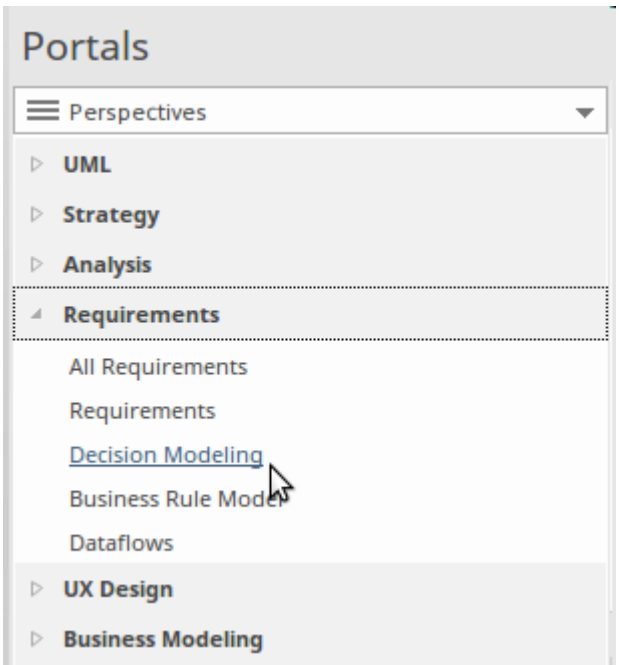
Every modeler will have their own preferences about the color scheme and style of the user interface and Enterprise Architect allows these to be set and saved for each user making the application more appealing and compliant with the way an individual user works. For example some modelers will want a dark color scheme and others will prefer a light or colorful scheme.



There is a range of options here including setting the position of the main window tabs and the size of the text in the notes window and much more. Setting the visual style will assist in personalizing the modeling environment and making individual modelers feel comfortable while maintaining consistent and rigorous models. For more information see the Visual Styles Help topic.

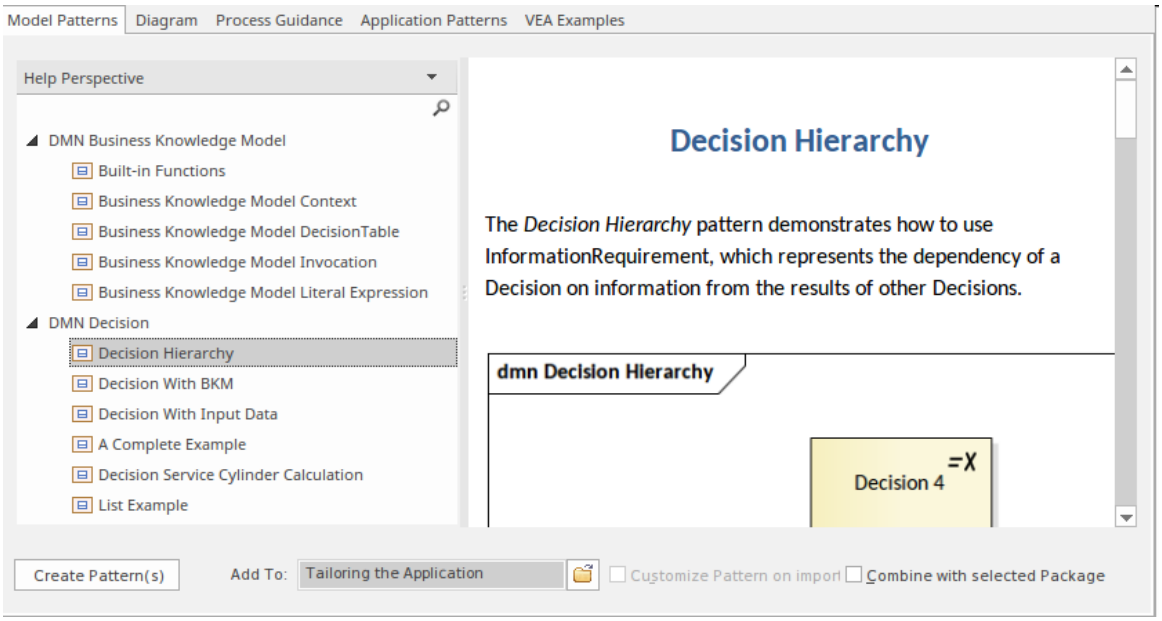
Selecting a Perspective

Enterprise Architect is a tool packed with features for a wide range of disciplines, methods, languages and frameworks. Perspectives provide a way for a user to select a facet of the tool that allows them to focus on a particular subset of the tool's features and facilities. The Decision Modeling Perspective provides a natural starting point for users focused on modeling decisions, but at any point if you decide to use other facilities in the tool you can simply change Perspectives and the tool will reshape itself to provide a focus on the selected area.



Selecting the DMN Perspective will change the tools to focus to the Decision Modeling facilities for example, the application will display a series of model patterns giving a user a jump start by being able to load a pattern for a standard model fragment or diagram. The 'New Diagram' dialog will also just display the Decision Requirements

Diagram type.

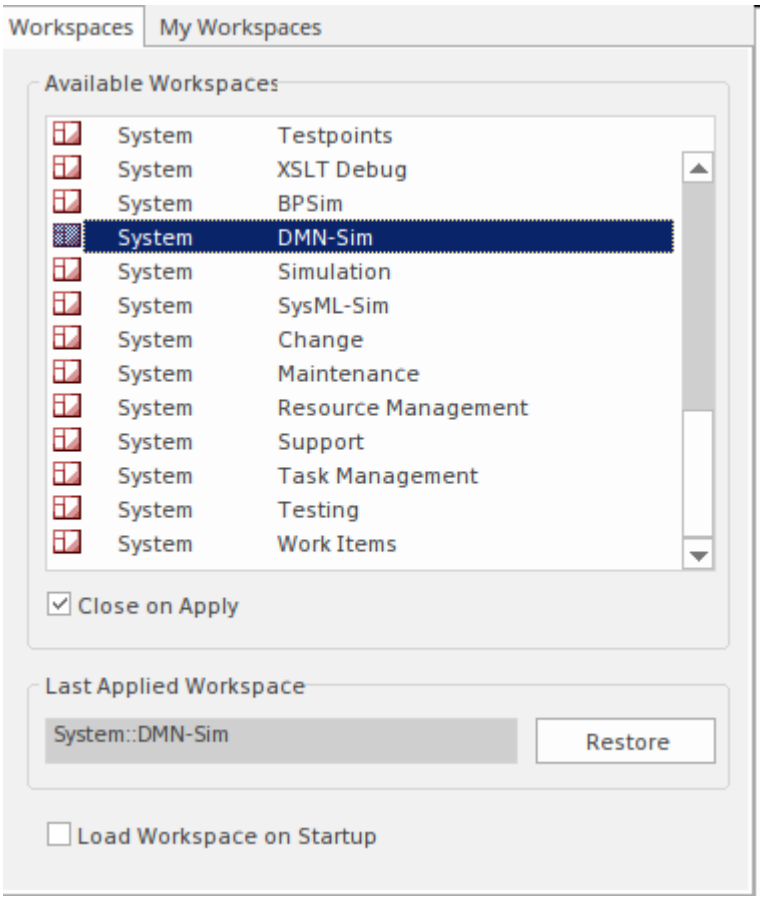


There is also the convenient facility for a user to create any number of their own Perspectives, adding sets of technologies to each Perspective. This allows a modeler whose primary concern is Decision Modeling diagrams to add other facilities such as strategic modeling, Requirements Modeling, Kanban diagrams and dozens of other useful diagramming and modeling mechanisms. For more information see the Model Perspectives topic.

Selecting a Workspace

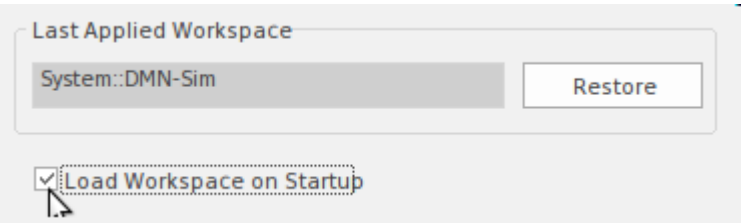
Enterprise Architect has a handy way of quickly changing the layout of the User Interface to facilitate particular tasks or ways of working. This is achieved by simply selecting a workspace that will change the visible windows and tools, to provide the most efficient way of working to suit the task.

For example, there is a workspace defined for Requirements Modeling, one for Use Case Modeling, and another one for Testing. We will select the DMN Simulation workspace as we will be using the simulation features in Enterprise Architect to run test executions of the models we create in this guide. You can also define any number of your own workspace layouts that you find useful, by opening windows and tools and positioning them in an arrangement that facilitate working on a particular task or set of tasks, and saving them. The Workspace Layout window is available by selecting the Workspace Item from the Application panel of the Start ribbon. For more information see the Workspace Layouts Help topic.



The dialog also provides the option to apply this workspace when the application is launched, which allows a modeler

who is working primarily on Decision Modeling to have all the tools at their fingertips without needing to set the workspace manually.



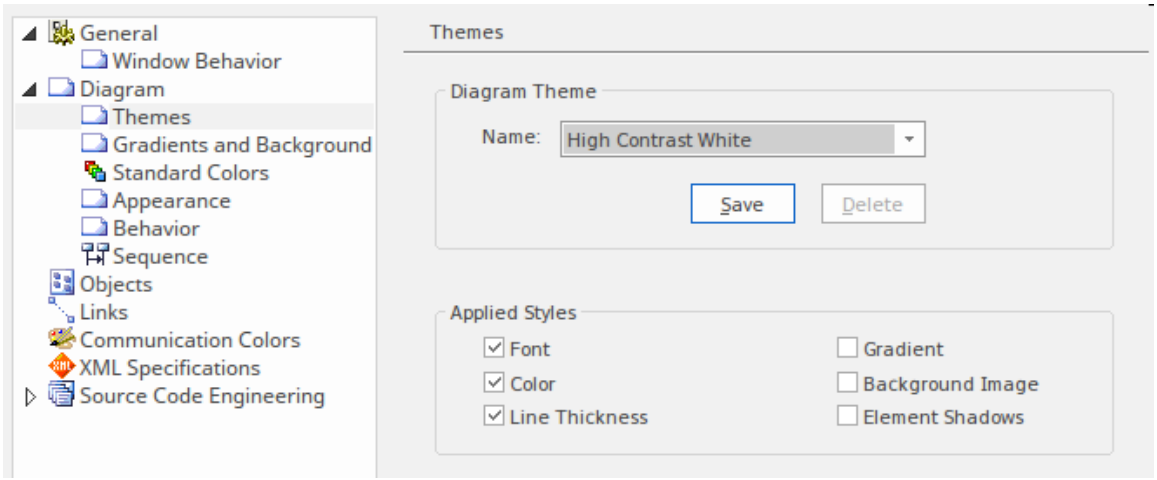
Setting Preferences

Enterprise Architect has a formidable set of preferences, some of which can be set for the entire repository and others for each user. These allow the application to be tailored to suit an individual modeler or an entire team. The preferences include options for:

- Windows - *styles, tabs, captions and status bar text.*
- Diagrams - *including themes, grids, gradients, backgrounds, styles and colors.*
- Elements - *colors, behaviors, display options.*
- Connectors - *styles, line width, directions.*
- Source Code Engineering - *general and language specific options.*

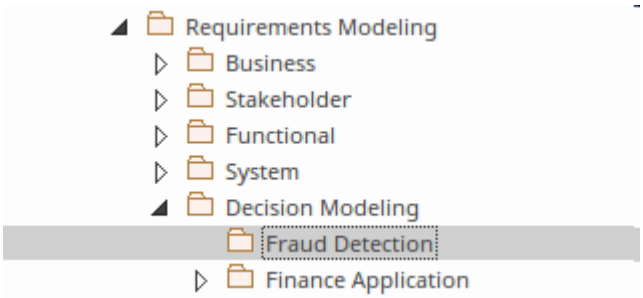
For more information see the User Preferences topic.

This illustration shows how diagram themes can be set and elements of the style can be specified, including fonts, colors, line thickness and gradients.


General
Window Behavior
Diagram
Themes
Gradients and Background
Standard Colors
Appearance
Behavior
Sequence
Objects
Links
Communication Colors
XML Specifications
Source Code Engineering
Themes
Diagram Theme
Name:
High Contrast White
Save
Delete
Applied Styles
Font
Color
Line Thickness
Gradient
Background Image
Element Shadows

Creating Diagrams Elements and Relationships

Once a model structure has been set up and the application has been tailored to suit your needs, including the selection of the DMN Perspective and an appropriate workspace, you are ready to start creating diagrams and elements. While it is possible to create an element without first creating a diagram it is common practice to begin by creating a diagram as a canvas for how the elements will be visualized. The first thing you will need to do is choose a location for the diagram in the Browser window. For example, you might be working on a project and have defined a Package called 'Fraud Detection' under the Decision Modeling Package. By selecting the 'Fraud Detection' Package you are telling Enterprise Architect that this is where you want a new Decision Requirements diagram to be inserted.

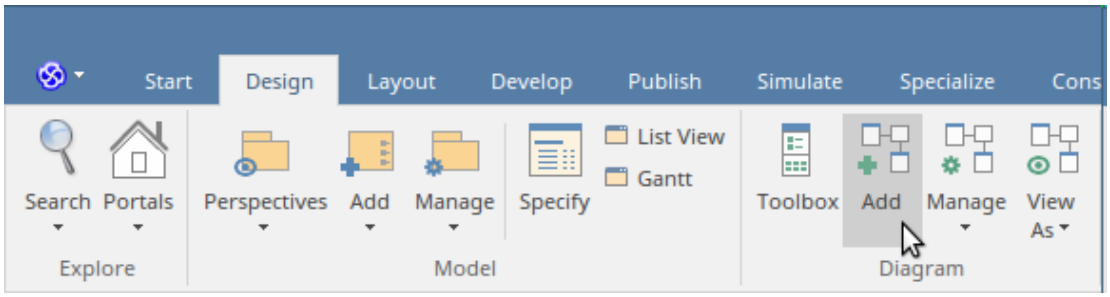


In the next few sections we will learn how to create a diagram and add elements, relationships (requirements) and artifacts that will define the model that describes the important decisions, their input data and knowledge sources.

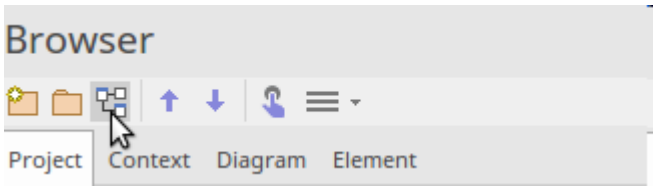
Creating and Viewing Diagrams

The Decision Modeling Notation is effectively a visual grammar, and while there are invisible syntactic items much of the grammar is expressed and viewed visually with diagrams. Thus it is important to be able to create and view the diagrams as these will effectively define the sentences that express the content of decisions. Enterprise Architect is a flexible tool and provides a number of ways of opening an existing diagram and creating a new diagram, including:

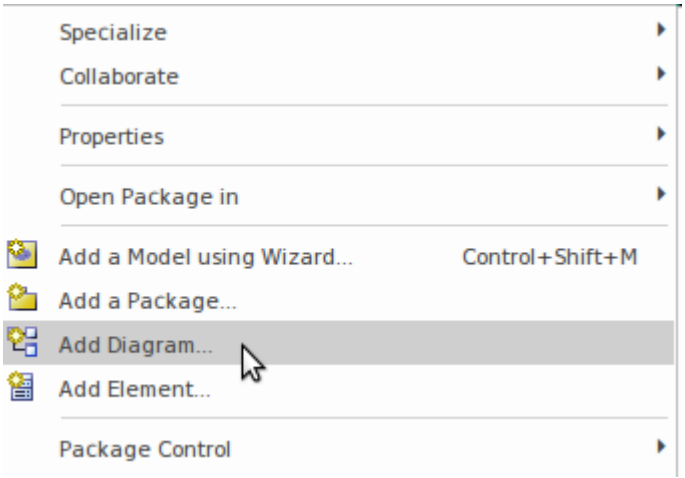
- Selection from the ribbons:



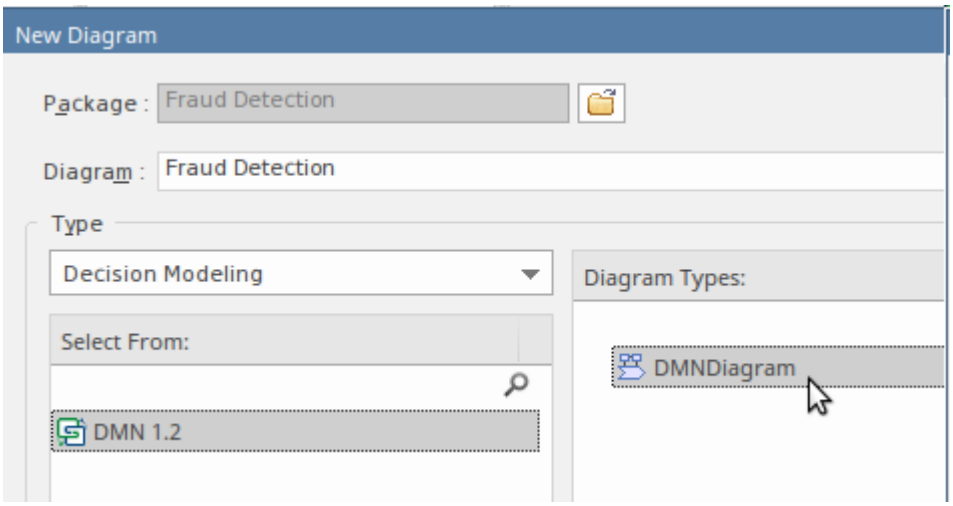
- Selection from the Browser window header bar:



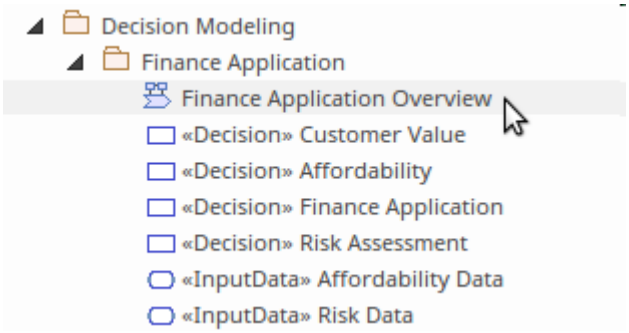
- Selection from the Browser window context (right click) menu:



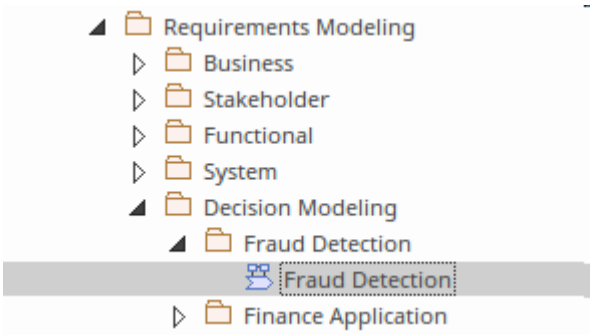
Regardless of the method you choose you will be able to select the DMN diagram type from the 'Diagram Types' panel of the 'New Diagram' dialog.



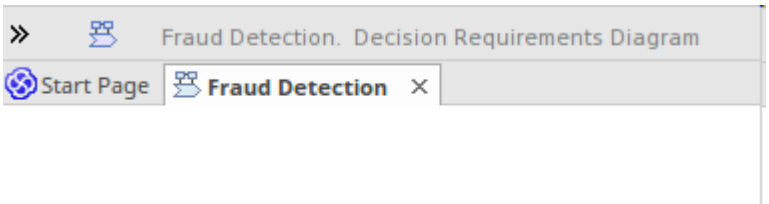
To open an existing diagram, you locate it in the Browser window and then double-click, or right-click and select *Open* from the diagram's context menu.



Let's continue on to create a Decision Requirements diagram (DRD) to represent the 'Fraud Detection' decision model. Select the Decision Requirements diagram as the diagram type and enter an appropriate name (note the name of the Package will be used as a default). Once you click on the OK button, a new (blank) DRD diagram will be created and opened and the DMN Toolbox will be displayed ready for you, or a member of your team, to create elements and relationships. As shown in this illustration, a new node will be added to the Browser window, underneath the selected Package, representing the diagram. The diagram can at any point be renamed, or moved to an alternative location, and properties of the diagram can be added or updated.



Enterprise Architect will create a diagram canvas with an invisible frame that represents the border of the diagram. The header information is contained at the top of the canvas and displays the diagram name and the diagram type. The diagram frame can optionally be displayed in documentation and other output as required



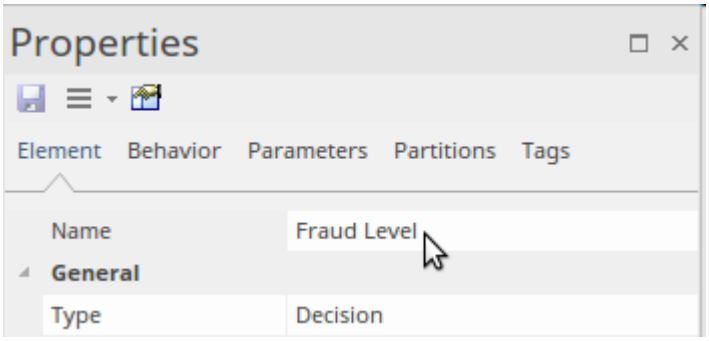
Adding Elements to a Diagram

With the new (or existing) diagram opened you are ready to start creating elements and relationships to describe the decisions. There are essentially two types of object that can be added to a diagram:

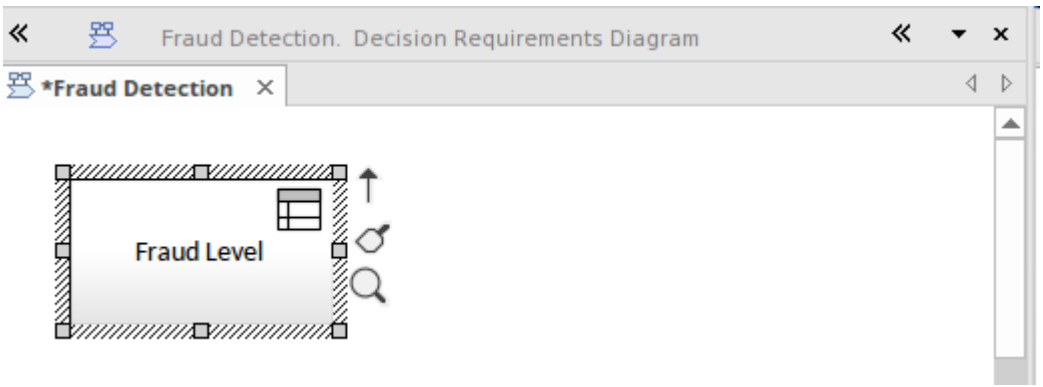
- New elements - *Created by dragging an item from the Toolbox and dropping it onto the diagram canvas*
- Existing elements - *Placed on the diagram by dragging-and-dropping an element from the Browser window*

If you are starting a new project and have just set up your repository, you will not typically have elements in the Browser window so you will make more use of the first option and create elements from the Toolbox. As your project progresses it will become more common to use the second option and drag existing elements from the Browser window onto the diagram.

We will create a new Decision, so we will drag and drop a Decision item from the Toolbox onto the diagram canvas. The element will be given a default name of 'Decision1'. Now using the Properties window, typically docked on the side of the diagram, change the element's name to 'Fraud Level' by typing over the default name 'Decision1'.



This will change the element's name in the Browser window and the diagram. Returning to the diagram you will see the newly added Decision with the name 'Fraud Level'.



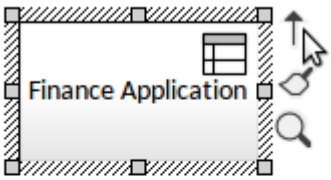
We could now use the same method to add any number of other elements, including other Decisions, Input Data, Business Knowledge Models and more. These other elements are all available from the Toolbox.

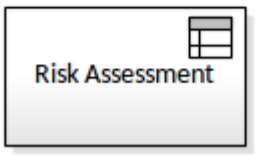
Adding Relationships to a Diagram

Once you have added two or more elements you can connect them with relationships, which provide the semantic glue between the different elements in the model. For example, an Input Data element is connected to a Decision element using an *Information Requirement* relationship. There are two primary ways that connectors can be added to a diagram:

1. Quick Linker - *an intuitive diagram device initiated by dragging a link between the Quick Linker arrow (at the top right of the element) and another diagram object.*
2. ToolBox Items - *connectors can be selected in the Toolbox and then dragged between two diagram objects.*

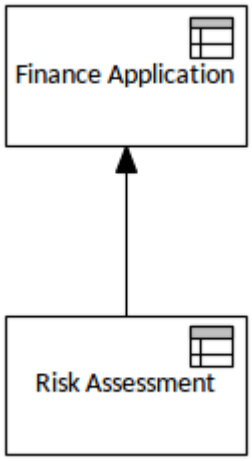
Either method will result in the specified connector being drawn between the two elements. Care needs to be exercised to ensure you are dragging in the right direction - a DMN Information Requirement connector shows that client A requires information from supplier B, so you drag from A to B; but the flow of information is from supplier B to client A, so the resulting arrow points from B to A. The same logic applies to the Knowledge Requirement connector.





Regardless of the method that is used the result will be an Information Requirement relationship connecting the two Decisions. The direction and style of the connector can be altered and any number of way-points can be added to route it differently as the model is developed. This diagram shows

the result of adding the connector, if a modeler were to inadvertently add the connector in the wrong direction it can be conveniently reversed by accessing options from the *Advanced* submenu of the connector's context menu.



Windows and Tools

We have already seen some parts of the user interface that can be used to create and maintain decision models, including the Browser window, the 'New Diagram' dialog and canvas, and the Perspective and Workspace facilities. When you select the DMN perspective Enterprise Architect loads the appropriate internal technology, effectively changing the tool into a Decision Modeling platform so that when you elect to create a new diagram you will only see the option to create a Decision Requirements diagram.

One of the most useful aspects of Enterprise Architect's implementation of the DMN standard is the ability to simulate Decision Models. This means that a Requirements Analyst or other stakeholder can perform trial executions of the model using test data to simulate what output a Decision Model would generate in a particular context. So, for example, in our *Fraud Detection* model a number of inputs to the Decision Model could be provided, and an output would be produced based on this input and the rules that had been configured.

Enterprise Architect provides two important windows for working with decisions; we will introduce both of these windows now and, as you will see throughout the Guidebook, these windows will be used extensively.

- The DMN Expression Window - used to define the expression in the form of Decision Tables, boxed expressions and other expression types

- The DMN Simulation Window - used to configure and run simulations

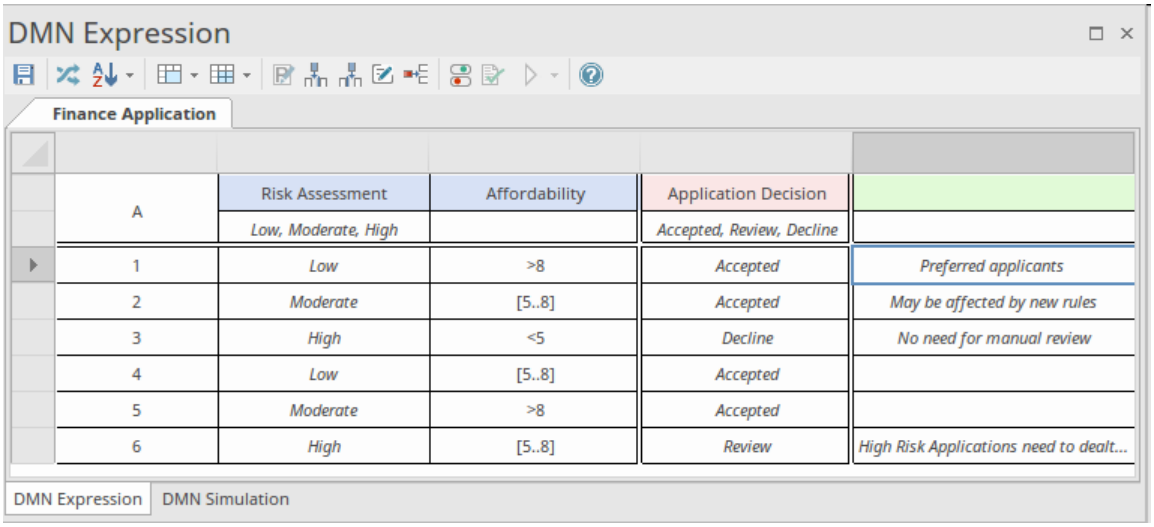
In addition there is a wide range of tools that can be used to facilitate working with Decision Models; we will have a look at some of these later in the Guidebook. These additional tools include:

- Document Generation
- Discussion and Reviews
- Traceability
- Relationship Matrices
- Diagram Filters
- Specification Manager

DMN Expression Window

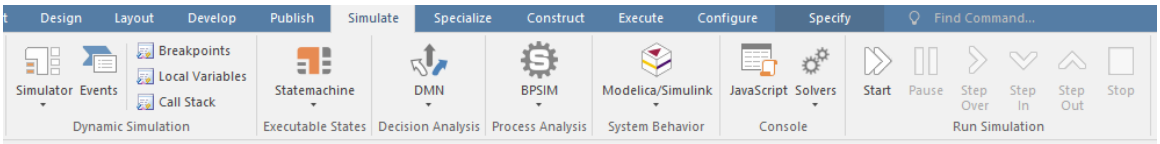
We have spent a lot of time introducing the DMN and describing the context for the models including the location in the Browser window and adding diagram elements. The expression window is the engine room or control center for the decisions and it is where the logic resides that effectively converts input information into a decision output based on the user defined expressions, which could for example be rules defined in a decision table. There is a validation facility that allows the rules defined in Decision Tables to be tested for correctness and coverage. It is also where input data is defined including Data Sets that can be created and reused to run the simulation using pre-defined input data allowing testing to be performed in a robust and repeatable

way.

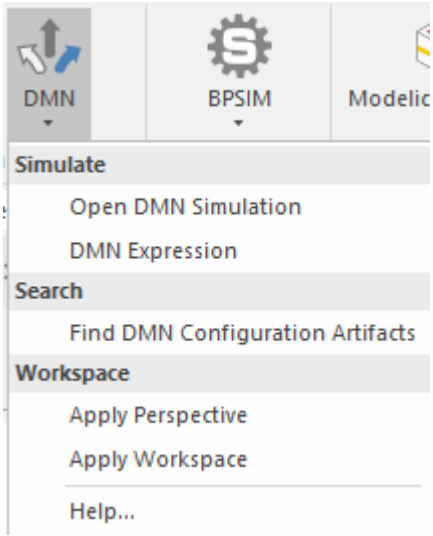


Simulate Ribbon

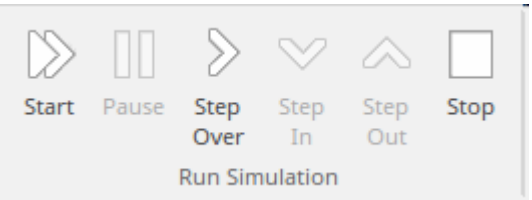
The Simulate ribbon provides a selection of tools for working with simulations, so here we see the Decision Modeling Simulation facility keep company with a number of other simulation tools, including Business Process Simulation (BPSim), Modelica/Simulink (SysML Pararametric) and StateMachine Simulation.



The DMN item on the ribbon has a drop-down menu that provides a number of facilities for working with Decision Modeling simulations. These include selecting a workspace and perspective and finding configuration items that represent model fragments with simulations that have already been configured.



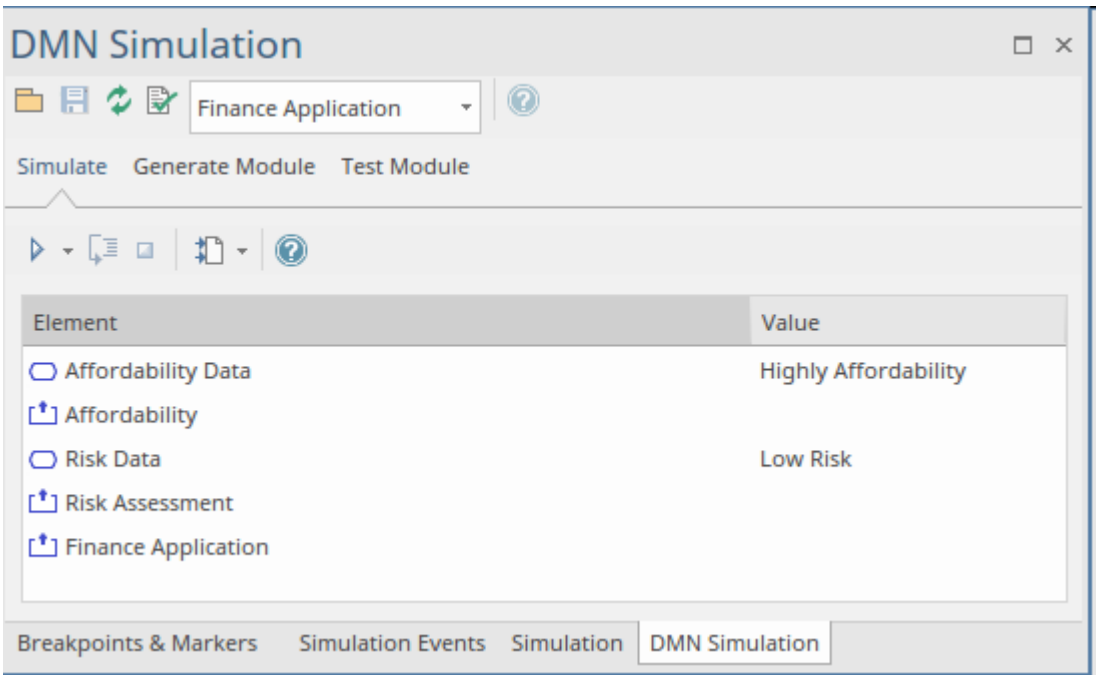
The ribbon provides a generic control panel for running simulations of any kind; this can be used when a DMN simulation has been configured and loaded. It allows simulations to be run to completion, paused, stepped through or stopped. This facility will be your friend as you develop Decision Models, or evaluate existing ones and need to understand what parts of the model have contributed to the final output. We will look at running simulations in more detail later in the Guidebook.



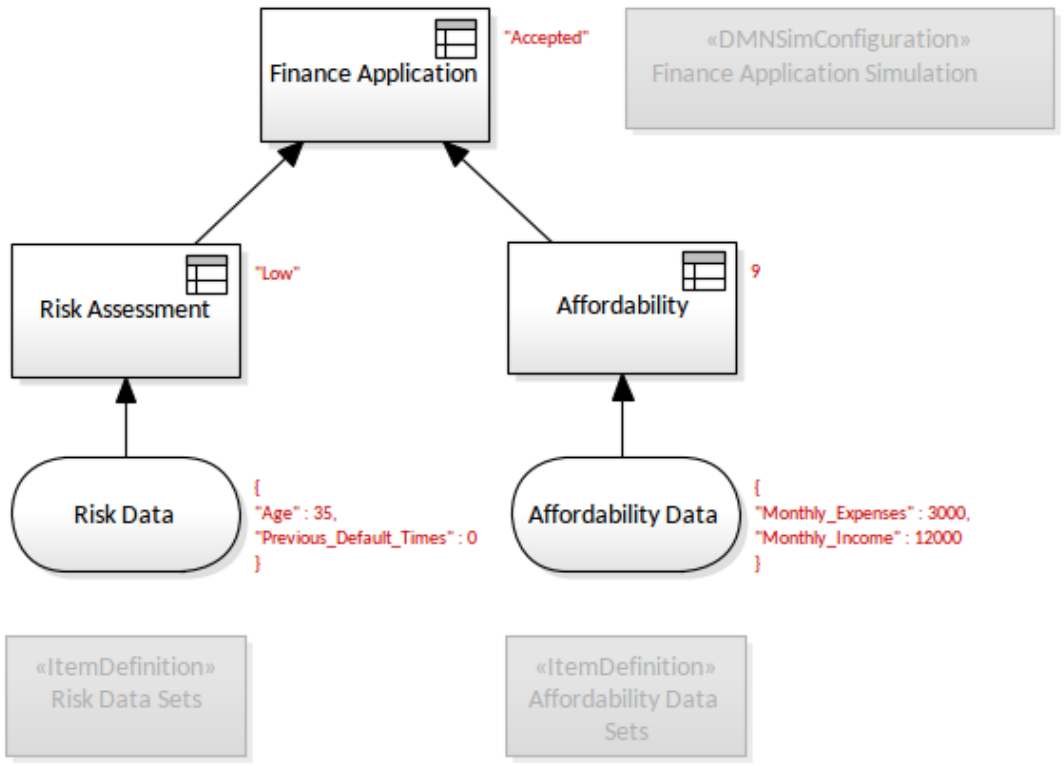
DMN Simulation Window

The Simulation window provides the facilities to configure and run the simulation including selecting what Data Sets should be used with each decision. The result of the simulation will be displayed in the current diagram in the form of annotations to the diagram elements. The simulation

can also be controlled from the Run Simulation Panel of the Simulate Ribbon and both control panels allow the simulation to be stepped through showing which rules in the decision expressions (including decision tables) are selected for the given input data. The simulation windows will become active when a Simulation Configuration is loaded and this can be achieved by double clicking the Configuration element in the current diagram.



The second diagram demonstrates the annotations to a diagram that are created when a simulation is run. The elements have the inputs and outputs annotated for each level of the decision graph and these can be seen to be bubbling up from the lower decisions in the hierarchy, up to the highest decision (the one that has only incoming but does not have any outgoing information requirements).



Other Tools

In the previous sections we have introduced you to the main windows used when working with the Decision Models, and in the topic before this one we introduced the main diagram used with Decision modeling, namely the Decision Requirements diagram. There is, however, a wide range of other tools that will be invaluable when creating and managing these models, which will be introduced later in the document. We introduced the Browser window when discussing setting up a repository structure but we haven't yet introduced the other tabs or views of the repository, which are useful to get a focused or 'keyhole' view of the part of the decision model that you are working on.

- Context Browser - provides a focused subset of the

elements in the project browser

- Diagram Browser - provides a list of the elements and connectors on the current diagram
- Details display (Inspector window) - provides a list of the important features, properties and relationships of the current element

Decision Model and Notation Overview

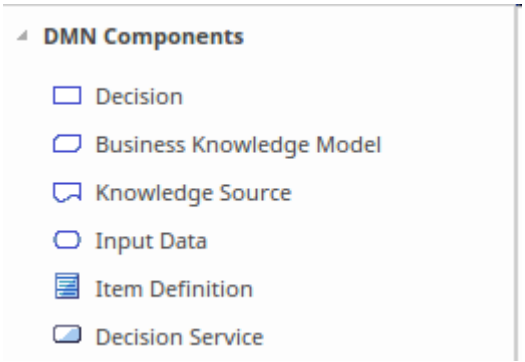
The fast pace with which our world is changing, the higher expectations of customers, their impatience and intolerance of poor services and the myriad of choices they have in a competitive digital world has led to organizations needing to be able to change quickly and efficiently and to understand the decisions that underpin their business models and the implementation of these models.

Decision Modeling has not been well recognized as a discrete discipline or science and business rules and decisions have been modeled in a heterogeneous way with little consistency within organizations or between them. Decisions have 'peppered' Business Process Diagrams in the form of cascading gateways making them complex and intractable to change and moving them further from the 'holy grail' of straight through processing. This situation has made collaboration and team work almost impossible and has resulted in poorly defined specifications, countless programming faults, frustration and ultimately unhappy customers.

This section addresses the questions of what Decision Model and Notation (DMN) is, where it fits in the context of other disciplines such as Business Process Modeling and Software Engineering, and what the characteristics of a good model are. The section also discusses the levels of usage and why and when to use DMN and gives a first

example that will help a newcomer to appreciate the benefits of the standard.

Enterprise Architect's pragmatic approach to modeling and the extensive set of facilities available to the Business Analyst and Software Engineer and others makes it a helpful tool as a repository for decisions and a platform for creating, managing, simulating, implementing and disseminating decision models and collaborating as a team to bring true and repeatable value to an organization.



What is Decision Model and Notation

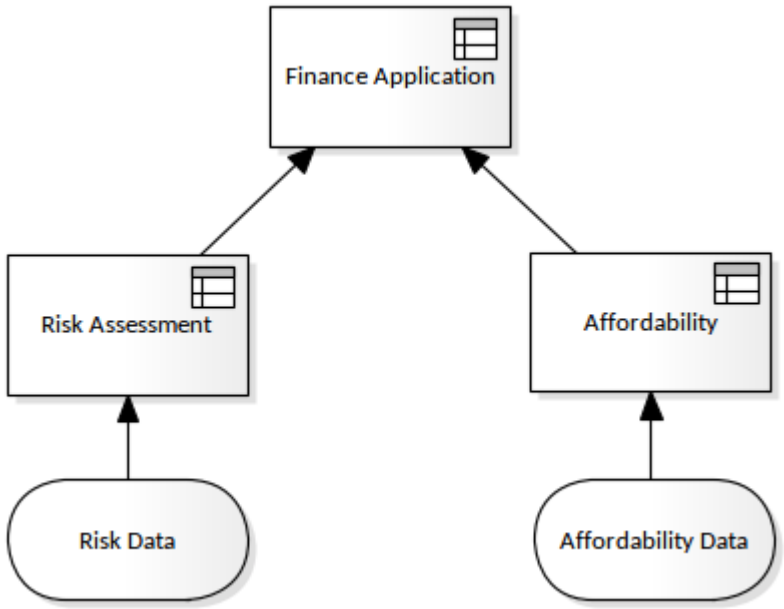
Decision Model and Notation (DMN) is a standard published and managed by the Object Management Group (OMG). It provides a standard approach for describing, modeling, simulating, reusing and implementing repeatable decisions within an organization or an initiative. It is also intended to facilitate the sharing and interchange of Decision models between organizations.

The modeling notation consists of a visual grammar that allows decisions and business rules to be documented in a way that makes them readable by both business and technical audiences, thus ensuring that decisions and rules are not misinterpreted. The resulting Decision model also provides a definition of how to evaluate the logic of decisions defined in Decision Tables, using the Friendly Enough Expression Language (FEEL).

The DMN defines two levels of the language, which neatly align with organizational roles:

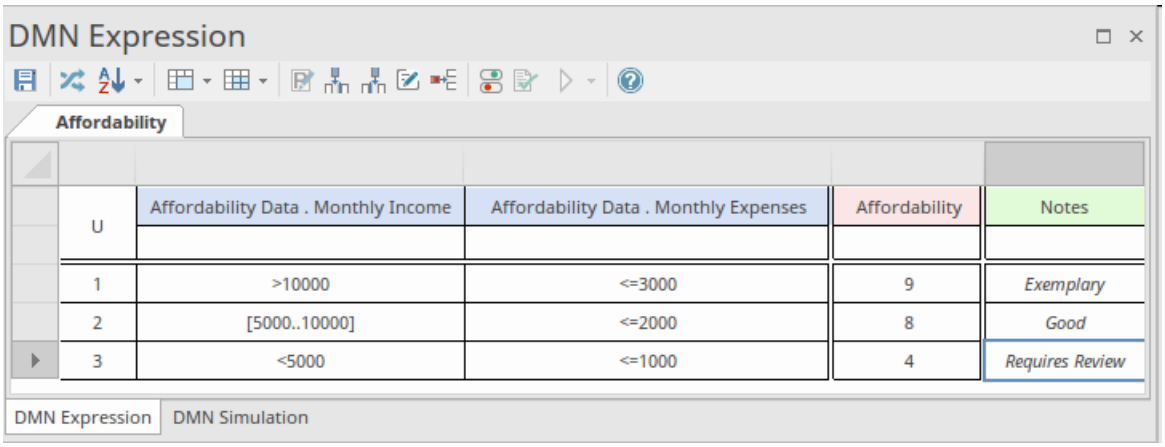
Requirements Level

Aligns with the business people who devise, analyze and define the decision rules. This provides a mechanism for the people who own the business to model the rules without the need for technical knowledge of how they are implemented.

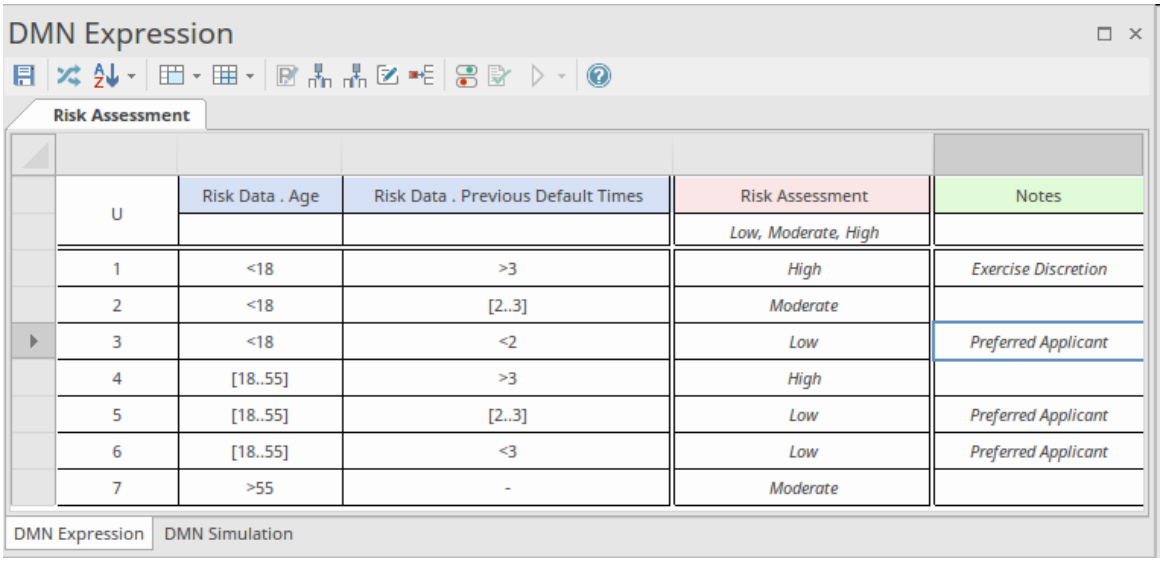


Logic Level

Aligns with implementation teams, who augment the definitions to include decision logic in the form of Decision Tables that can be maintained by the business staff, or expression logic managed by technical staff.



In this example we see that the main Decision has two inputs from other Decision Tables that are processed first and that contribute to the overall decision about whether the application will be accepted or not. This table shows the Risk Assessment Decision Table.

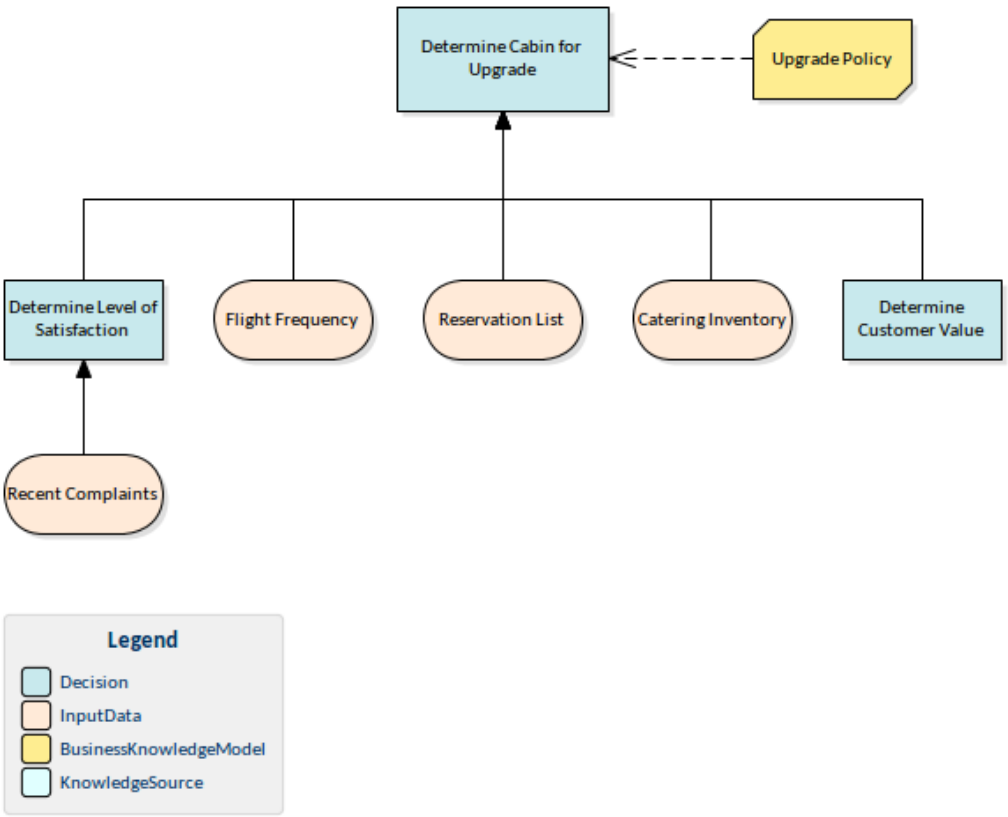


U	Risk Data . Age	Risk Data . Previous Default Times	Risk Assessment	Notes
			Low, Moderate, High	
1	<18	>3	*High*	*Exercise Discretion*
2	<18	[2..3]	*Moderate*	
3	<18	<2	*Low*	*Preferred Applicant*
4	[18..55]	>3	*High*	
5	[18..55]	[2..3]	*Low*	*Preferred Applicant*
6	[18..55]	<3	*Low*	*Preferred Applicant*
7	>55	-	*Moderate*	

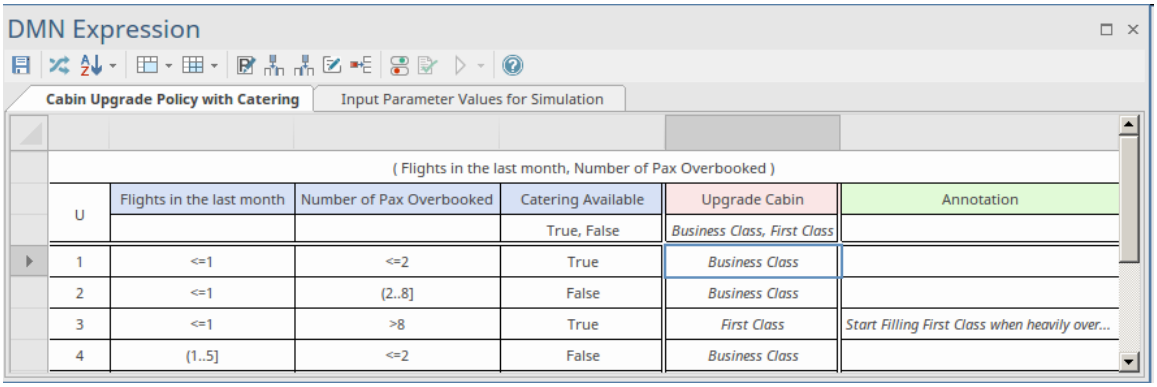
A First Example

Imagine you are an Airline Reservation Officer working at the check-in counter for a busy domestic airline. Getting the aircraft off on time is critical as delays can result in fees applied by the airport controllers, and possibly having to fly at a lower altitude, increasing the cost of fuel and the payment of other penalties. Not to mention the inconvenience for passengers who could miss important meetings and who - next time they fly - might choose another airline.

Imagine you are in the process of checking passengers at a busy domestic airport. A message from the supervisor appears on your screen saying that the economy cabin is overbooked; you have to upgrade some passengers to Business or First Class - but which passengers should be chosen? A decision has to be made, but what factors should be considered? The busy check-in officer is not in a position to be able to make this decision. They could be helped by viewing some of the factors they should consider that have been recorded in a Decision Model using a Decision Requirements diagram.



This is helpful but the busy check-in officer would still need to weigh-up all the factors and make an unbiased decision. Should a disgruntled passenger be given priority over a Gold level frequent flyer, or should the fact that a particular passenger is connecting to an international flight take precedence. These 'rules' can all be recorded in a Decision Table, making it clear which passengers should get an upgrade and to which cabin, Business or First class. This will make it much easier to make the decision and the rules can be formulated, agreed upon and checked for consistency back at head office.


DMN Expression

Cabin Upgrade Policy with Catering | Input Parameter Values for Simulation

(Flights in the last month, Number of Pax Overbooked)

U	Flights in the last month	Number of Pax Overbooked	Catering Available	Upgrade Cabin	Annotation
			True, False	*Business Class, First Class*	
1	<=1	<=2	True	*Business Class*	
2	<=1	(2..8]	False	*Business Class*	
3	<=1	>8	True	*First Class*	*Start Filling First Class when heavily over...*
4	(1..5]	<=2	False	*Business Class*	

The table is divided into columns and rows. There are two types of column: inputs that are required to make the decision and outputs that are the result of applying the rules.

This is very helpful but still requires the busy check-in officer to be able to source all the information required to find the right row in the Decision Table. Even if all this information were available, an inappropriate decision could still result from human error in selecting the wrong row in the table

Fortunately the Decision Models can be automated and generated to programming code that can be executed by an application. So our busy check-in officer would not need to do anything or make any decisions as they are checking in the passengers; if a particular passenger was entitled to an upgrade, the reservation would have been automatically altered and the upgrade would be visible on the check-in screen. The only thing the Airline Reservation Officer would need to do is provide the welcomed news to the passenger - 'Mr Travelealot we are delighted to provide you with an upgrade to First class on your flight to New York this morning'

It is common for the rules that govern the upgrade decision to change. For example, the marketing department might

decide that they want to reward passengers that travel on long-haul flights. The Decision Requirements diagram can be altered to include the new input, the Decision Table modified, tested and simulated and the programming code regenerated. Once the changes have been pushed through to the airport systems, magically the right passengers will be upgraded. The check-in officer could still view the Decision Tables during a training and briefing sessions so as to understand the rules. Many of the implementation engines would also provide a business friendly explanation as to the basis of the decision so the check-in officer could explain to the passenger the reasons for their upgrade.

Levels of Usage

The creators of Decision Model and Notation intended it to be used in a variety of ways by different organizations and also within the same organization. The division of the models into Requirement, Logic and Implementation levels pre-empts usage scenarios for:

- Modeling Human Decision-Making
- Modeling Requirements for automated decision-making
- Modeling automated decision making

Some text books and white papers talk about these as levels of maturity in the field of decision making, but fail to explain that there are many situations - even in highly automated systems - where it is important or even imperative for human beings to be making the final decisions, particularly in areas of human safety; these will not be automated, but are required to be described for governance and regulatory reasons.

Modeling Human Decision-Making

DMN can be used to model enterprise, organization or initiative level decisions that are performed by staff rather than computer systems. Typically the decisions are described at quite a high level and the rules are commonly written in a natural language but can be described using more formal mechanisms such as Decision Tables. At this level it is useful to model how knowledge relates to decisions using Business Knowledge Models that capture

specific areas of business knowledge and its applicability to one or more decisions for example a set of Standard Operating Procedures. Knowledge Sources can also be modeled that describe the sources of Business Knowledge for example a Standard Operating Procedures Manual. This type of modeling can be both descriptive and prescriptive.

Modeling Requirements for Automated Decision Making

When DMN is used for modeling the requirements for an automated decision making system it is important that the inputs and outputs are defined and the models are developed to the level of fidelity needed for an automated system.

Modeling and Implementing Automated Decision Making

Enterprise Architect is a full-life-cycle tool that can be used to model decisions for *Automated Decision Model* systems and can generate programming code in a number of languages, which allows the decision requirements and their accompanying definitions to be enshrined in implementation code. This is a useful mechanism that leads to great productivity and ensures that there is an unbroken chain between the highest level thinkers in an organization and the final automated systems that process the decisions and determine their output, all within the one tool.

Why Use Decision Model and Notation

There are many reason for using DMN at an Enterprise or Initiative level that will result in both business and technical value. Many benefits are described in the next topic, but the most compelling are described here:

Reduced Complexity

As discussed earlier, currently business rules and decisions are commonly located in a multitude of places and in a wide variety of formats. They are often found in Business Process diagrams that are overly complex, with cascading sets of Gateways that describe the outcomes. The diagrams are commonly unwieldy and do little to reveal the inputs, business knowledge, authorities or the logic of how the decisions are made.

Automation and Execution

As the Business Analysts define the decisions, inputs and business knowledge, and the technologists elaborate the logic in the form of Expressions and Decision Tables, potentially creating Decision Services, the models can be used to generate programming language code that can be executed to automate the decisions and make them available to runtime systems.

Agility and Responsiveness to Change

As a result of separating the Decision models from the Business Process models, and creating an implementation and automation pipeline, the business and technology teams can respond at record speed to business change. These models then become a platform that facilitates agility and business responsiveness.

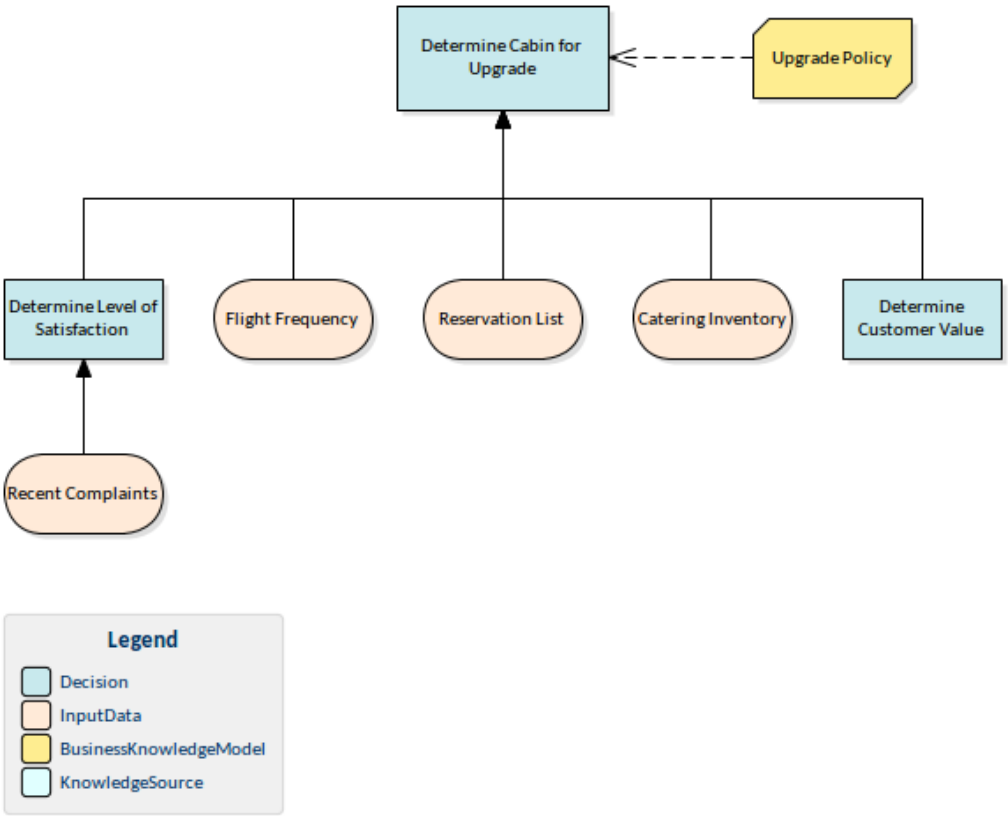
When to Use Decision Model and Notation

Decision Model and Notation (DMN) can be used in a wide range of places, from modeling high-level strategic decisions down to making specific decisions about hardware devices. This section describes some usage examples that will help you to understand when DMN can be used and what elements are available in the Decision models.

- Decide on the level of discount to offer a customer
- Decide which customers should receive upgrades to the First Class cabin
- Decide on the system access level to give a new employee or contractor
- Decide on what pages of a dynamic Web Site to display for a given visitor
- Decide on when to send an incoming aircraft into a holding pattern
- Decide on when a stocked item should be ordered, and how many units should be ordered
- Decide when to sell a stock or asset

Decide if a customer should receive an upgrade to Business or First class

To ensure optimal profit levels are achieved for each flight segment an airline typically overbooks all available cabins using closely guarded algorithms. As the time approaches scheduled departure the level of overbooking is reduced. This at times results in more tickets being sold than available seats, particularly in the economy cabin. To resolve the issue the airline will typically offer upgrades to economy passengers at check-in. The decision to upgrade a passenger should be based on commercially important factors and is a canonical situation where a Decision Model can help in creating a repeatable, unbiased and commercially valuable result.



Enterprise Architect can be used to create the Decision Requirements Model, Expressions and Truth Tables can be added to define the logic, and implementation code can be

automatically generated from the model.

Decide on the level of discount to offer a customer

Business personnel that work close to the front line of an organization will be familiar with the expression that the 'Customer is King'. Maintaining existing customers and acquiring new ones is critical to the viability of a business. Providing discounts to customers is an effective incentive mechanism to 'sweeten' a sale and provide a competitive price in a landscape dominated by competition. Leaving this decision to a busy sales representative will often result in discounts being given without any business reason or worse discounts not be given resulting in a sale going to a competitor and the potential of losing the customer. A decision model can provide the repeatable and commercially correct choice of which customers should receive a discount and the percentage or amount of the discount.

Decide on the system access level to give a new employee or contractor

Most organizations understand the importance of their staff in the overall success of their business. They also know the importance of providing the staff with the right tools to perform their roles, including the right level of access to the various software applications they use to carry out their

work. The process of on-boarding personnel is already complex enough and it is not feasible for a technology officer to be able to determine the list of applications and the level of access required.

Decide on the when and how many units of a stocked item should be ordered

Manufactures operate in a highly competitive environment and producing high quality goods while at the same time keeping costs to a minimum is critical to a successful and sustainable business. The decision as to when to replenish stock or component parts and how many to order is determined by a number of factors. Getting it wrong and under ordering can have an impact on a production line or sales bottom line, over ordering and having stock under utilized is expensive and has an impact on cashflow. Taking the example of a production line the

Decide on what pages of a dynamic Web Site to display for a given visitor

Web sites have come a long way from their early incarnations at the dawn of the Internet and now they are just one aspect of an overarching digital strategy. Individual customers or classes of visitors can be identified and using a vast amount of available data from prior interactions to customer relationship information, such as purchasing

records it is possible to determine the most effective set of pages to display to them and to create interest trails that will lead them to a purchase. It is not possible to do this manually and these rules have typically been built into the server and client side scripts that control the dynamic aspects of the web site.

Creating a decision model that not only describes the inputs and rules but incorporates them into an overarching digital strategy including: social media, business architecture and customer relationship management would provide competitive advantage for any organization. The website would then become a dynamic decision driven engine that created content based on a sophisticated engine where the rules were visible and part of a well understood and articulated model resulting from business and technical collaboration.

Decide on when to send an incoming aircraft into a holding pattern

Air services are the commonest and most in demand form of transporting people and goods both around the globe and quite locally. There are many companies competing to provide these services, and whilst the number and size of air terminals are increasing, they are still a limited resource and need extremely careful management of the flights using them, and the people and goods passing through them. Of the millions of decisions that have to be made and coordinated daily, consider one: when should a flight be

directed to fly in a holding pattern before landing? One answer might be immediately if a dangerous situation exists on the ground, but it is localized and expected to be resolved soon. Otherwise the flight would be redirected to another airport (where they might also be directed into a holding pattern just to fit them in to the normal schedule), or, on the other hand, if a small factor means immediate landing is not convenient. But how small a factor, and what convenience, can justify that action? Having posed the question, other decisions are required: What actions must be taken to deal with the delayed landing? And what consequences must be weighed and accepted? How many flights might it be necessary to put in a holding pattern, given the answer to the original question: When? A Decision Model would be an extremely valuable tool to manage even this one situation, let alone all the others arising in the daily operation of just one airport.

Decide when to sell a stock or asset

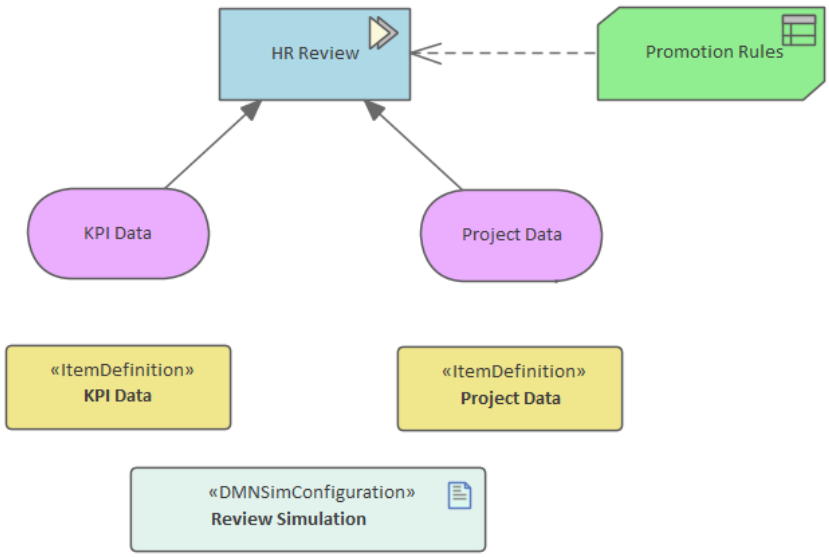
The stock market is a volatile institution, and decisions to buy or sell shares often seem to depend on nothing but whim. However, there are actually good and solid metrics and factors that can be assessed and cross-linked to indicate whether a buy, hold or sell position should be taken. A Decision Model is an excellent tool for weighing the answers to vital questions concerning the management of the company for which the shares are held, the indices of profit and loss or profit against investment, whether you are

searching for a weaker stock to divest to enable purchase of stronger stock, or seeking just to reduce exposure to weak stock or the point at which to capitalize on a strong one. An investment advisor or stock broker would also pay particular attention to investor profiles as well as company performance metrics before recommending a position or portfolio to an investor. This might make use of two Decision Models, the output of one model (perhaps Risk Factor or current Risk Exposure) being input as the answer to a question in the other model.

An Example Decision Model

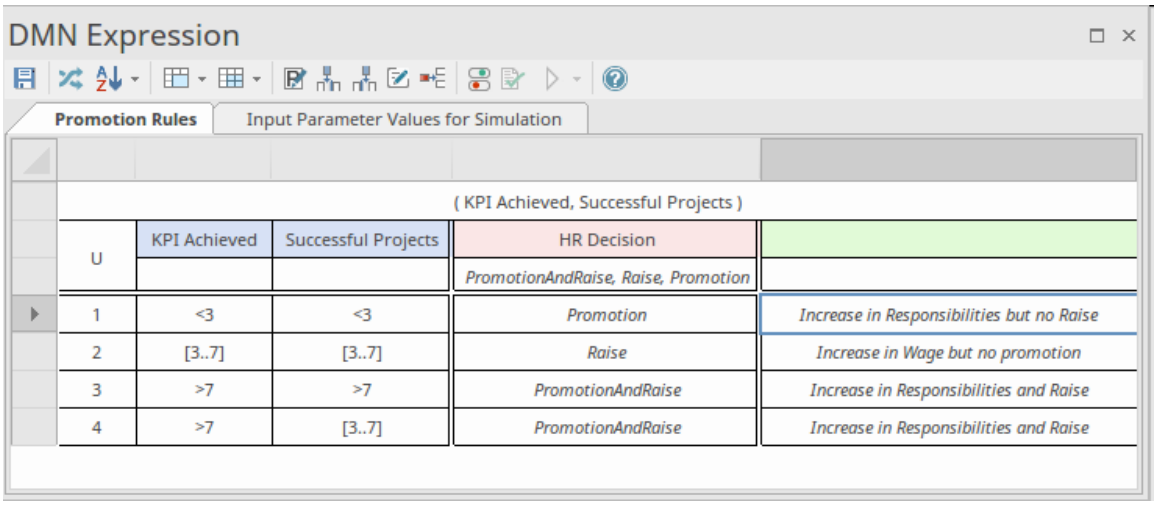
This example Decision Model uses the basic and common elements of the Decision Model and Notation language to demonstrate and exemplify a business decision that we all have probably been interested in at some point - the HR Review decision resulting in a possible wage raise or promotion.

The HR Review Decision element does not itself define rules but rather is of type Invocation, which calls the services of the Promotion Rules element. This is a Business Knowledge Model (BKM) that can be reused in different contexts. The BKM defines the rules for promotion in a Decision Table that is stored internally, and when it is called it passes the resulting output back to the Decision element.

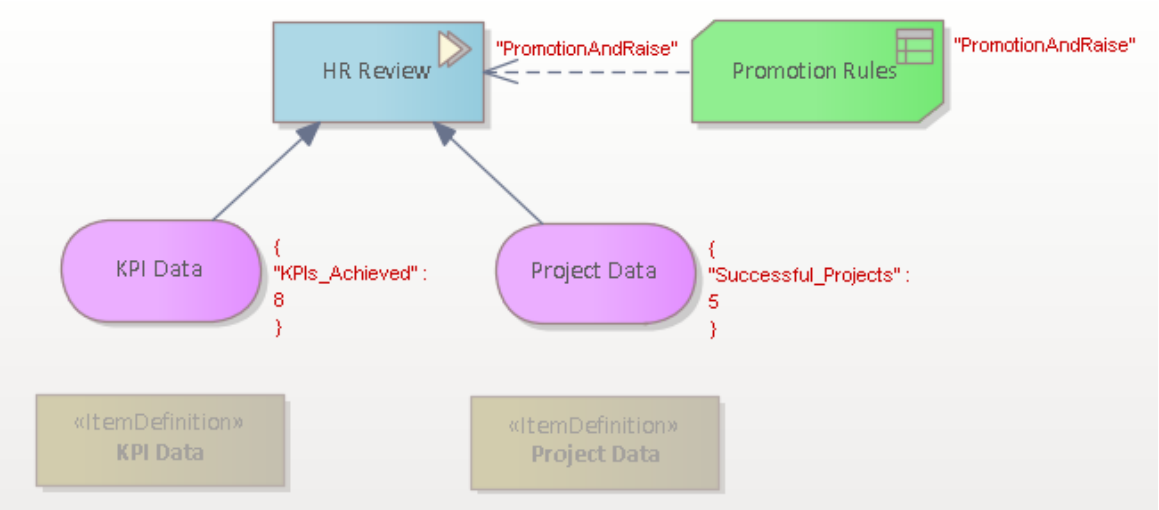


The Decision element has two inputs, namely Key Performance Indicator (KPI) Data and Project Data - this data is used as input to the decision. Enterprise Architect has a helpful mechanism for defining different data sets that can

be used to simulate the Decisions and effectively allow business and technical users to perform 'what-if' analysis as part of pre-production testing or in-production support analysis. The Decision Expression editor helps you to create and modify rules in an easy to use table that colors the input and output columns and has a number of built-in features that make it easy for business and technology users to work with decisions. Any number of Annotation columns can be inserted to create additional documentation to explain the expressions.



Enterprise Architect is a full life-cycle tool, and once the Decision Models have been created and tested they can be generated to implementation code, ensuring that the decisions that the business defines are enshrined in code. As part of the testing process the tool provides a simulation engine that allows the models to be visualized as though they were in a production system. Any number of data sets can be defined and used with the simulation to test the decision logic and thus avoid errors occurring in production systems.


HR Review
"PromotionAndRaise"
Promotion Rules
"PromotionAndRaise"
KPI Data
{
"KPIs_Achieved" :
8
}
Project Data
{
"Successful_Projects" :
5
}
«ItemDefinition»
KPI Data
«ItemDefinition»
Project Data

Benefits of Decision Model and Notation

There is a wide range of benefits that a team or organization can derive from using Decision Models and the associated notation to record the important decisions for an initiative, project, division or enterprise. The most compelling of these benefits is probably the ability to collect and visualize the rules in a single model and to see how they relate both to each other and to the inputs, business knowledge models and knowledge sources that are used to determine the outputs. There is a wide range of other benefits, including the ability to generate the implementation of the decisions in the form of programming code directly from the models. Each organization will doubtless find innumerable benefits from using both the Decision Models and the notation; this section lists some of the core benefits, including the ability to:

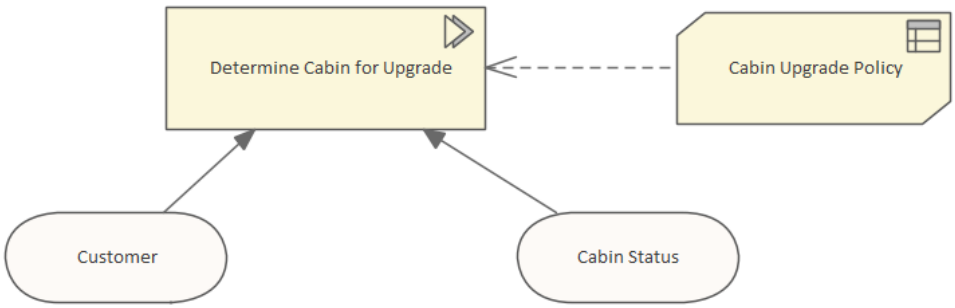
- Visualize the decisions and rules
- Simplify Business Process diagrams
- Visualize inputs and analytics
- Communicate shared understanding
- Facilitate collaboration
- Discover opportunities for automation
- Incorporate information into Architecture models
- Create a single source of 'decision' truth
- Simulate decision models

- Generate programming code
- Articulate with other models
- Check consistency and correctness
- Find gaps and overlaps in specifications
- Discover errors in specifications
- Test the models
- Create reusable libraries of decisions

Visualize Decisions and Rules

Business Rules and Decisions are typically scattered across an initiative's landscape, both at a specification and an implementation level, making the processes of managing them very difficult - if not intractable.

Using Enterprise Architect, the decisions can be collected together and managed in a common language and format, and visualized in a variety of ways.

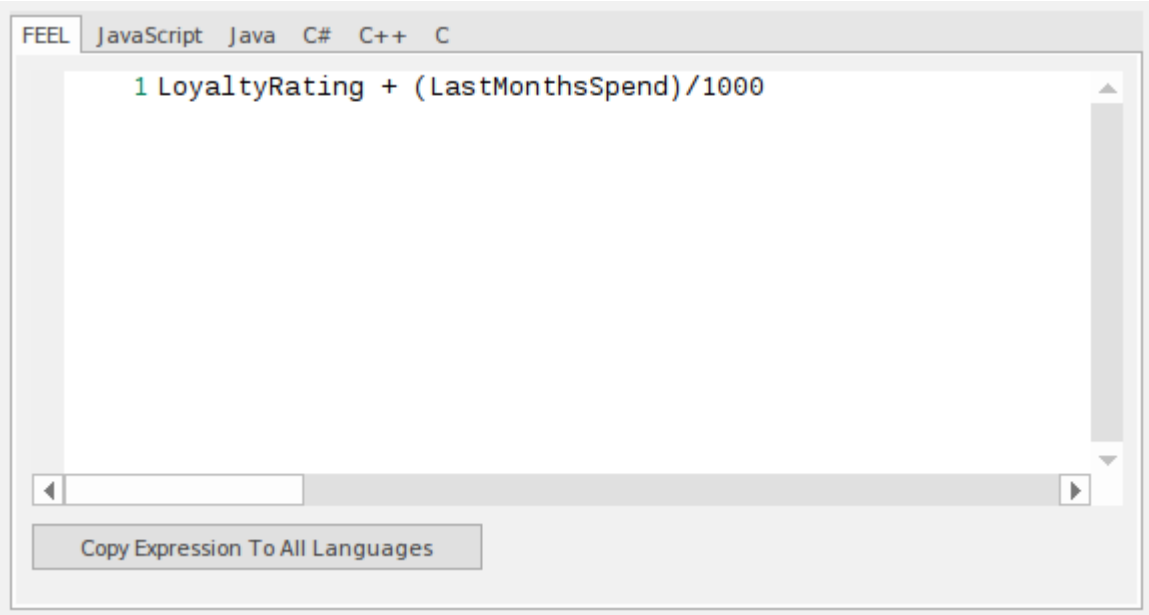


There are a number of simple ways to visualize the logic of a decision that make the specification of the logic easy for an analyst or non-technical stakeholder to specify, read or change. The most compelling of these is probably the Decision Tables that express the logic in a set of intersecting columns and rows. Typically the columns define the inputs and outputs, and the rows list the possible input values and are organized into rules that map these discrete values onto discrete output values. Enterprise Architect also allows the decision table to be inverted such that the columns articulate the rules and the rows list the inputs and outputs.

Another method of defining the rules, which will be more useful for simple situations, is using an expression language.

Enterprise Architect allows you to define the expression in a number of implementation languages including:

- Friendly Enough Expression Language (FEEL)
- JavaScript
- Java
- C#
- C++
- C



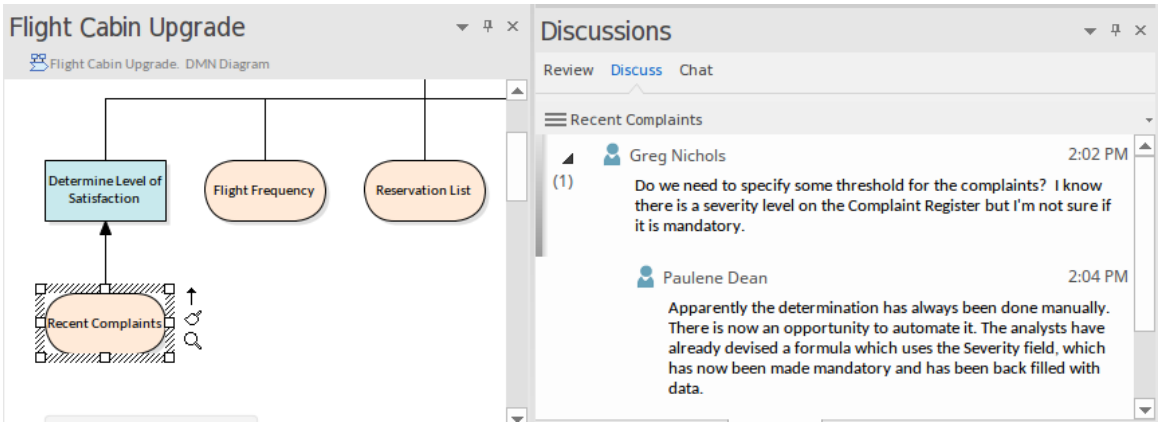
Facilitate Collaboration

Decisions are not developed in isolation, but rather require collaborative effort, often starting with some type of strategic intent and ending with an implementation of the Decision Model as a manual business process or automated in an information system. Either way, the realized decision process can be executed hundreds or, in the case of a computer system, millions of times a day.

Enterprise Architect is a team collaboration platform that has sophisticated tools to ensure that all the team members who need to contribute to the creation, development and maintenance of the models can collaborate in the same model, regardless of their role in the team, where they are located, or how they connect to the model. These tools include:

- Reviews - where models and expressions can be analyzed and criticized
- Model Library - a collection of documents and web resources that can be viewed
- Element and diagram discussions - ability to communicate about elements and diagrams with modelers and business users
- Chat - an in-model chat for immediate conversations
- Model Mail - an in-product communication tool allowing messages to reference model elements and diagrams
- Model Calendar - an in-model calendar for important team events

This diagram shows the Discussions facility that can be used to facilitate collaboration between teams that work in different buildings, or even in different countries and time-zones.



When clients use a Cloud Server environment, anyone can view and contribute to the models from anywhere, including from in-built browsers on these devices:

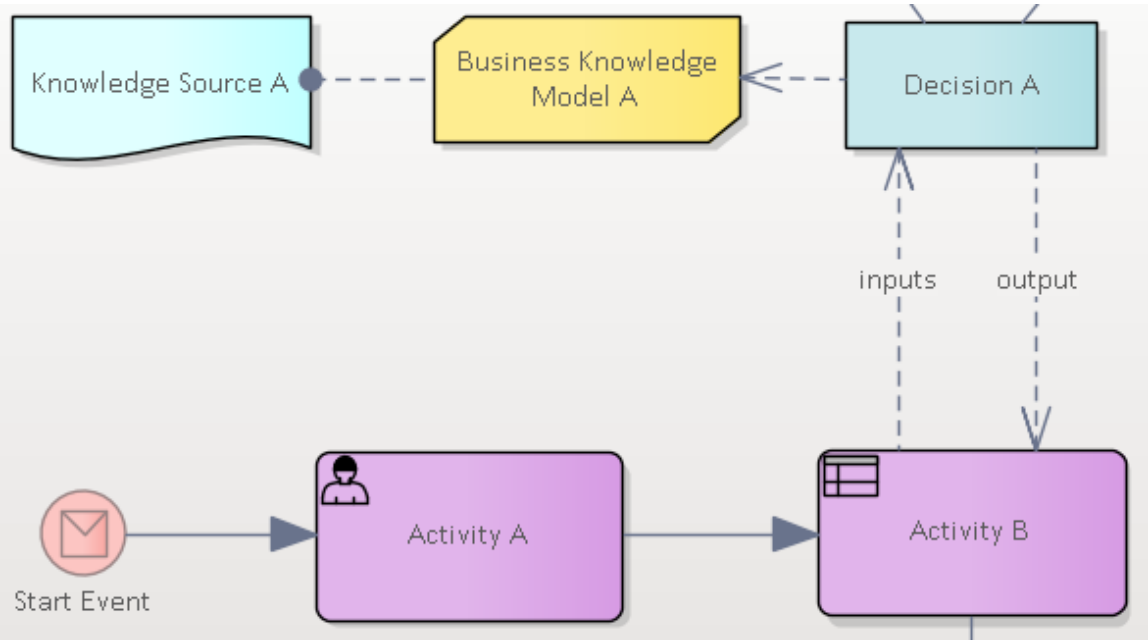
- Smart Phones
- Tablets
- Notebooks

This means that all levels of users from strategic thinkers down to implementation and support staff can access the models and collaborate wherever they are and using whatever device they have at hand. So a business line manager could be at the airport and open their smart phone and participate in a discussion about the data inputs to a particular decision or provide information about the definitive source for a given Business Knowledge model. A software engineer could be traveling home on a fast train and use a tablet to review some implementation details or generated programming code.

Simplify Business Process Diagrams

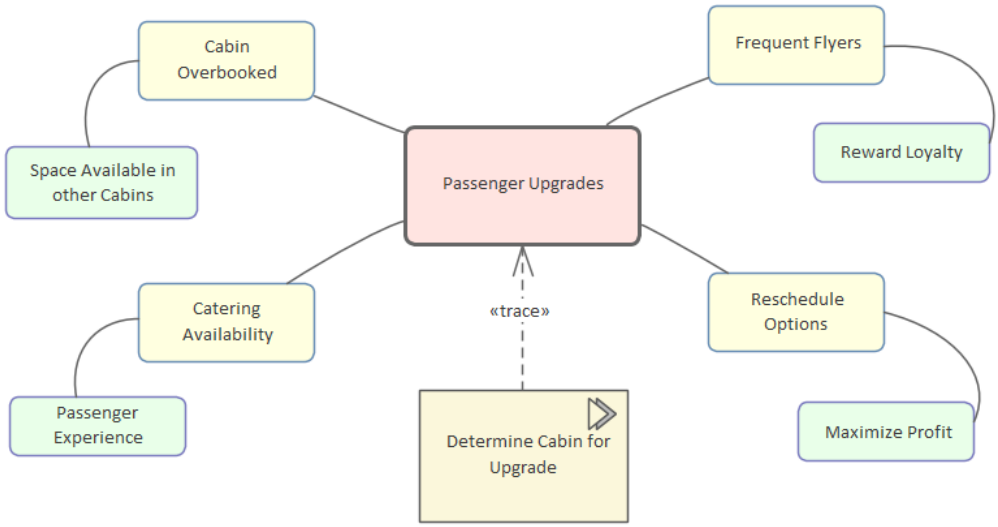
Business Process Models either created using the Business Process Model and Notation (BPMN) standard or the Unified Modeling Language (UML) Activity diagrams typically contain a myriad of decisions in the form of cascading gateways. These introduce unnecessary complexity and inhibit a reader's ability to understand the intent of the processes. They work against the principle of straight- through-processing and make the processes large and often unwieldy resulting in the attenuation of their value to both business and technical audiences.

Using Enterprise Architect to model both the Business Processes and the Decision Models means the process diagrams can be simplified, streamlined and created in a way that complies with the principle of straight-through-processing. The complex decision logic can be removed and an element can simply refer to a separate but connected Decision Model all conveniently located in the same repository.


Knowledge Source A
Business Knowledge Model A
Decision A
inputs
output
Start Event
Activity A
Activity B

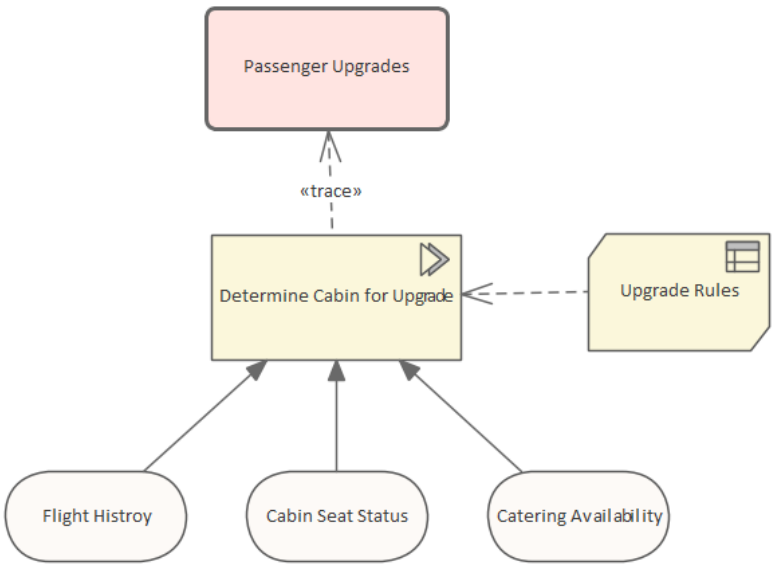
Communicate Shared Understanding

The decisions and rules that an organization makes will often shape the various ways the organization is perceived by customers and suppliers. These decisions are not created by a single person but are the result of many people's work, including strategic thinkers, Managers, Business Analysts, System Analysts, Software Engineers, Testers and others. Typically these groups work in different locations using different tools and only communicate with each in meetings or by email. The decisions are typically recorded in meeting notes or emails and are developed and implemented by a process of 'spider-web communications' where there is little trail of their origin.



Enterprise Architect is a collaboration platform that makes organizational and project information available to all team members, from the strategists who conceive the need for the decisions to the engineers and support staff who implement them and ensure they are working correctly. The need for a decision is often identified during a workshop, and the

initial ideas can immediately be recorded in Enterprise Architect using a Mind Mapping diagram or meeting notes.

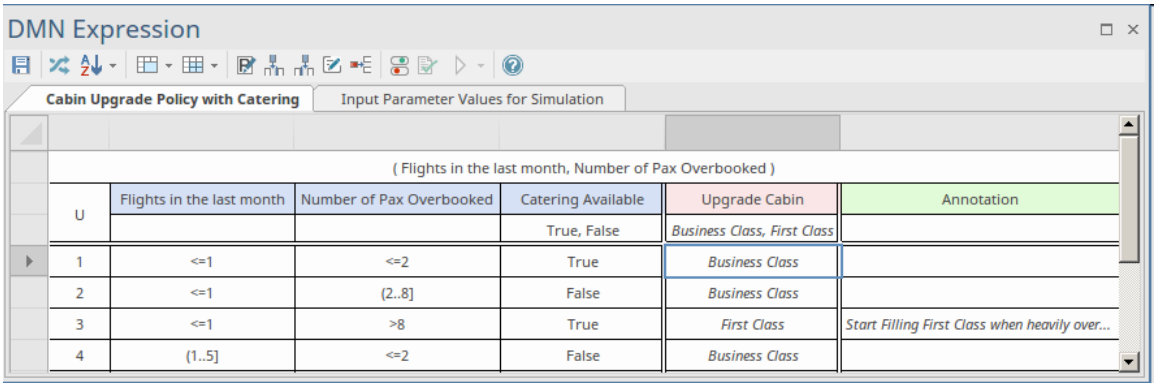


A Business Analyst can then refine the ideas and create a Decision Model that describes the decision and the inputs that are required to make the decision. This can include Business Knowledge Experts and Authorities that are the source of that knowledge. The decisions can be traced back to topics in the Mind Mapping diagram, allowing traceability back to the strategic level, and the Business Managers can also see clearly how their ideas are being analyzed and described.

Discover Opportunities for Automation

The decisions and rules that an organization applies are typically created and implemented over many years, and are often buried below the surface, almost requiring the business analyst to work as an archeologist to find them. The rules are built into work processes and job descriptions, written down in rarely-viewed manuals, and partially defined in programming code or database procedures. All this makes it difficult for the technologist who is trying to automate processes and make an organization efficient, because they cannot see the rules.

Using Enterprise Architect to define the decisions and rules in a Decision Model not only allows them to be visualized but enables opportunities for automation to be clearly understood and the opportunities subsequently implemented. The articulation of the Decision Models with the Business Process Models means entire blocks of processes can be removed and replaced by an automated decision-making solution. The result will often be directly observable by an organization's customers, who will enjoy reduced waiting times, greater flexibility, more accurate processes and, potentially, reduced costs for services.



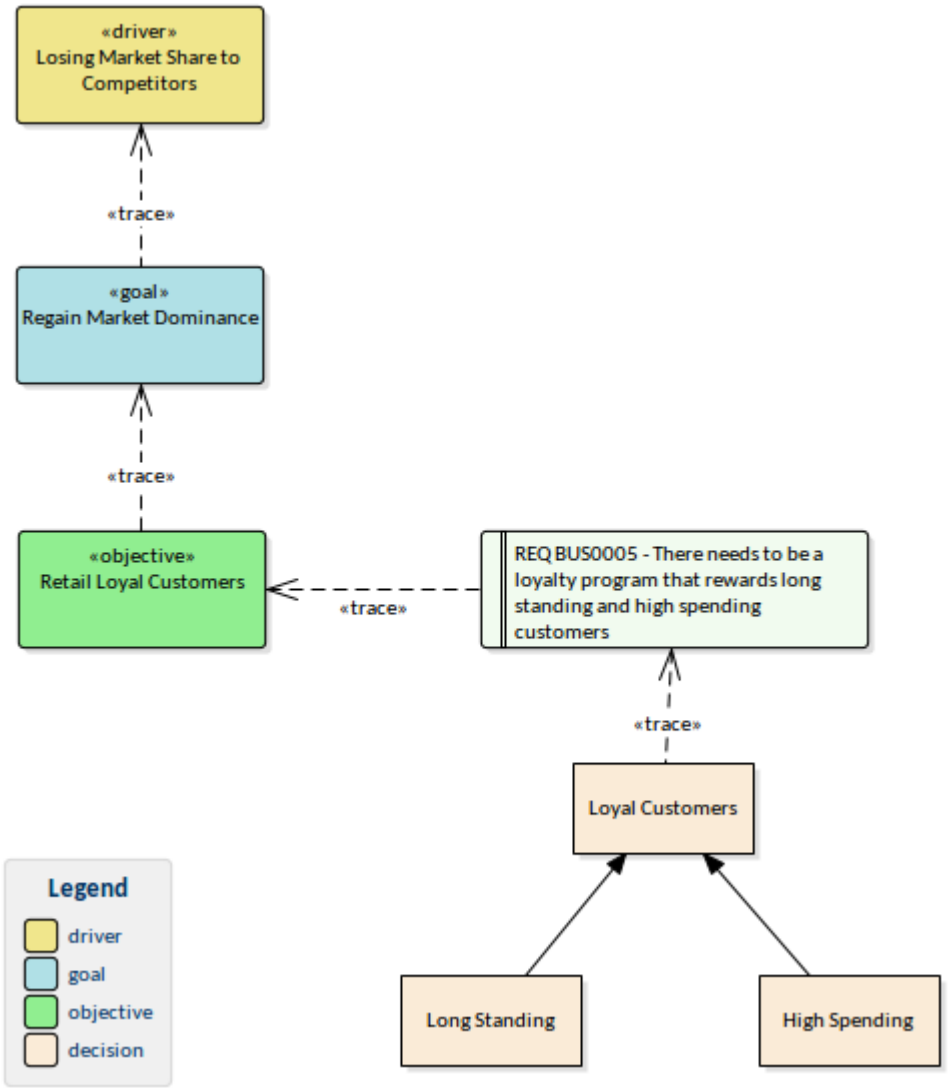
(Flights in the last month, Number of Pax Overbooked)

U	Flights in the last month	Number of Pax Overbooked	Catering Available	Upgrade Cabin	Annotation
			True, False	*Business Class, First Class*	
1	<=1	<=2	True	*Business Class*	
2	<=1	(2..8]	False	*Business Class*	
3	<=1	>8	True	*First Class*	*Start Filling First Class when heavily over...*
4	(1..5]	<=2	False	*Business Class*	

Incorporate into Architecture Models

Architectures are created to guide an enterprise or initiative from a baseline (current) state to a target (future) state. They typically encompass a wide range of architectural levels and types from Business through to Technology Architectures and translate notions such as goals, drivers and capabilities into roadmaps that define the direction that the enterprise or initiative will take to reach its target state and as a consequence its goals. Decisions can be built into the architectural models allowing them to be visualized as part of the architecture. These decisions will typically be defined at a high level but will capture the strategic and solution level thinking that would be otherwise lost in a textual document.

Enterprise Architect is quickly becoming the industry's tool of choice for both Enterprise and Solution level architecture and being able to build the decision models into the architecture models will ensure that there is traceability from the low level implementations right back to the design and architecture levels including requirement levels.

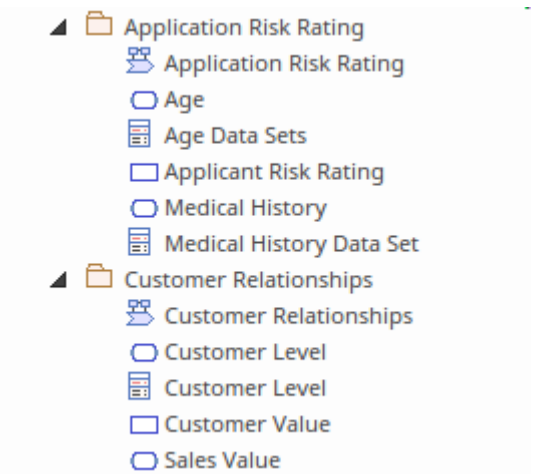

«driver»
Losing Market Share to Competitors
«trace»
«goal»
Regain Market Dominance
«trace»
«objective»
Retail Loyal Customers
«trace»
REQ BUS0005 - There needs to be a loyalty program that rewards long standing and high spending customers
«trace»
Loyal Customers
Long Standing
High Spending
Legend
driver
goal
objective
decision

Create a Single Source of Truth for Decisions

One of the things that has plagued organizations and resulted in many errors and inconsistencies and acted as a barrier to flexibility and agility is the fact that business rules and decision have traditionally been managed in a plethora of places. These range from spreadsheets, to strategy documents, mind maps, requirements documents, database models and programming code.

With DMN and Enterprise Architect there is the possibility of having a single source of truth and being able to manage these all important pieces of corporate information in a single Repository, which is searchable, can generate both documentation and implementation code.

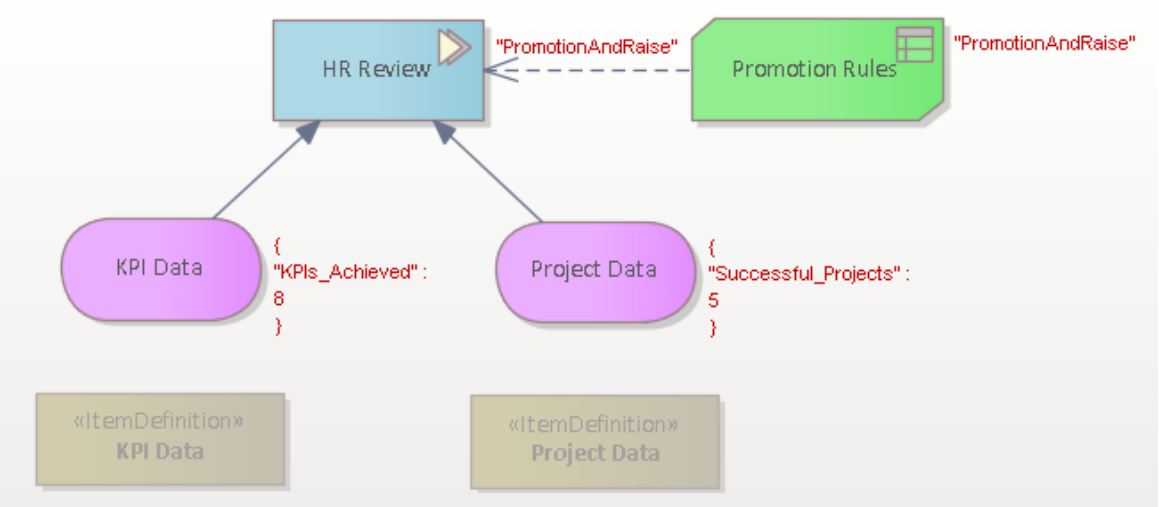
Having the decisions in a single repository solves the problem of managing networks of decisions that create a complex fabric that underpins the business value offered by the organization.



.

Simulate Decision Models

Simulations can be seen as a 'glamorous solution' of Decision Modeling, but beyond their instant appeal is the fact that they provide an effective means of visualizing, understanding, analyzing and communicating the intent and meaning of the models. The simulations can be run on the models as they are developed, before they are put into production, after structural model changes, or after rules or example data sets have been updated. The simulation facility is fundamentally a mechanism of running trial model executions for the entire model, or individual decisions within a complex model, allowing a team or an individual stakeholder to view the inputs, visualize the execution path, and analyze the intermediate decisions and outputs for a given model or fragment of a model. Simulations are also useful to perform what-if analysis and to check how new data sets in the domain will affect the decision outputs.

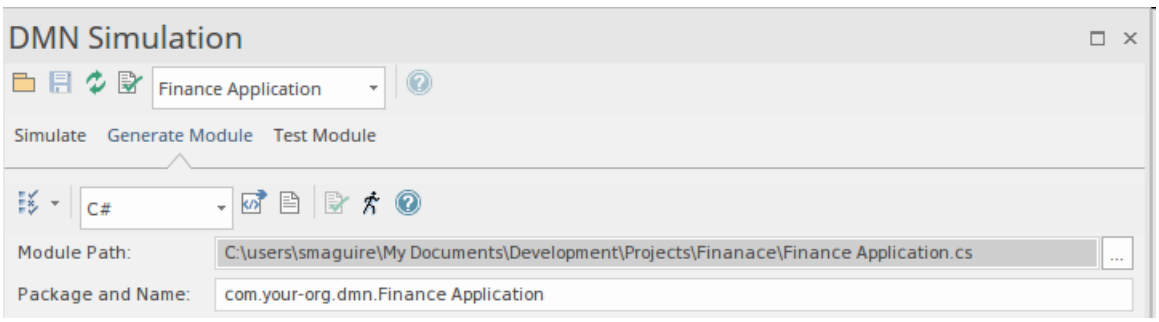


Test Decision Models

One of the compelling things about models is the ability to be able test a model before it is implemented or put into production. As explained earlier in this guidebook the cost of mistakes is high with some errors resulting in catastrophic outcomes, for example in aviation or safety systems. The Decision Modeling features in Enterprise Architect allow testing to be conducted in the form of simulations, the tool aslo has a number of useful testing features where test cases can be created and test runs recorded.

Generate Programming Code

One of the most compelling advantages of using Decision Model and Notation in Enterprise Architect is the ability to automate the decisions directly from the Logic level, allowing Decision Tables, Expressions and other definitions of the rules to be generated to a number of programming languages. This feature propels decision modeling into the digital age and closes the deep crevasses that have existed between Business and Technology teams. The implementation teams no longer need to consume document based definitions of the business rule in a particular domain, but rather receive a fully worked model that has been tested and specified down to the required level of detail. Moreover, the business teams no longer need to create the aforementioned documents that are time consuming and are typically out of date before the ink is dry. There will be occasions for technical teams to make some refinement to the rules for efficiency purposes or to assist with difficult logic, but generally this would be done with the sanction of the business. In these circumstances the simulations could be rerun with the same data sets to ensure that the results are what the business owners are expecting.

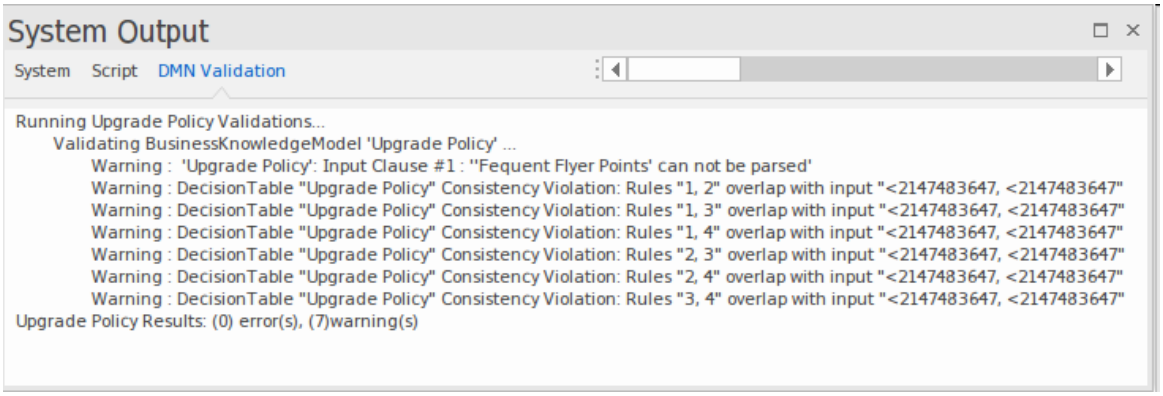


Check Consistency and Correctness

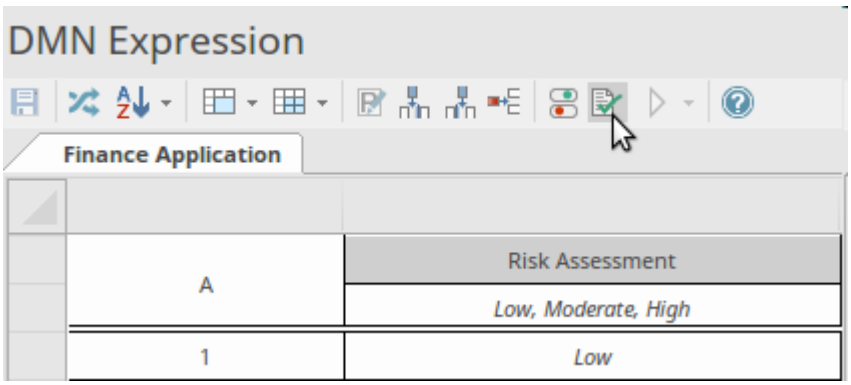
Even with well planned and executed testing many of the faults in software centric systems are found at run-time by users. This is also true with decisions and business rules and it is not until, the wrong information is sent to the wrong customers or a security device allows access to a request that should have been blocked, that we know that something has gone wrong.

Decisions and the logic that describes them is often complex and hard for humans to grapple with. This situation is compounded because decisions are commonly connected together in a graph creating an interwoven fabric where one decision depends on three others which in turn depend on others and so on.

Enterprise Architect implementation of the DMN standard makes it easy to visualize and collaborate together defining and automating a set of decisions. The rules can be defined and displayed in the expression editor manifested in Decision Tables that resemble spreadsheets which will be familiar and reassuring to business analysts and other business stakeholders. The rules defined in these tables can be checked for consistency and correctness alerting the analyst to problems of missing or overlapping conditions and other issues.



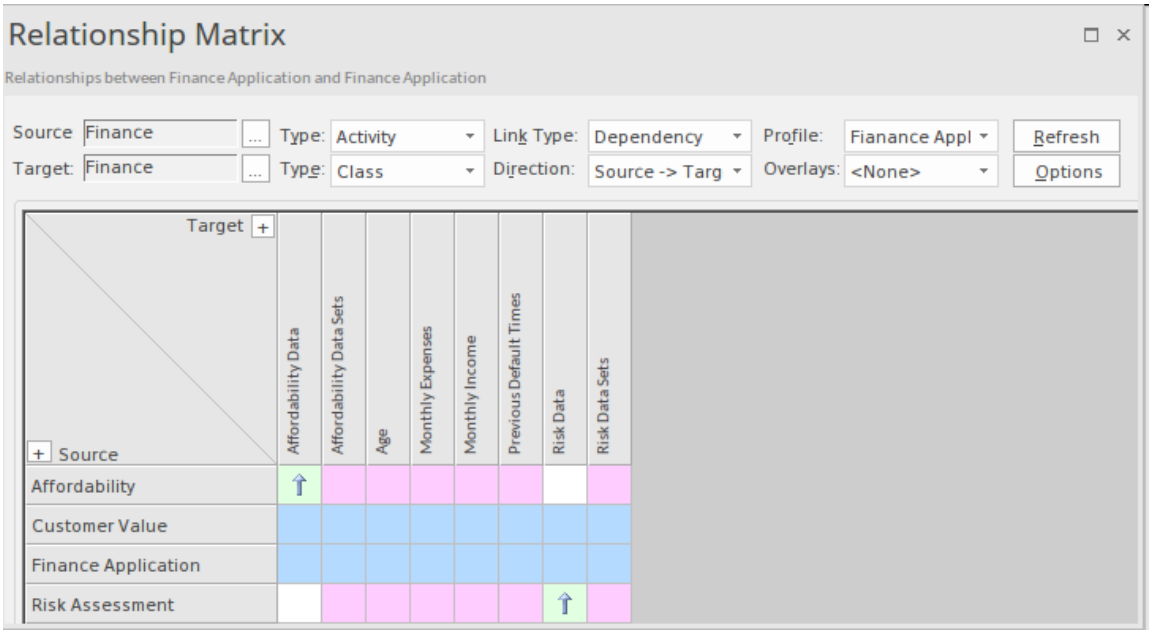
The tool that performs the validation is conveniently built into the Decision Table window and means that the problems can be resolved early in the process long before they get enshrined in programming code and reach the customer.



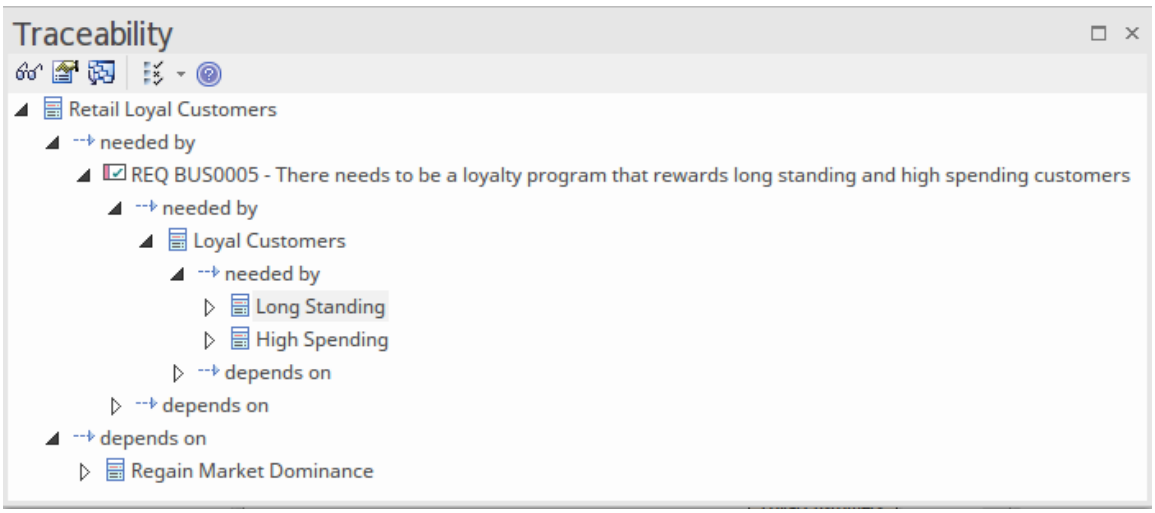
Find Gaps in Specifications

Creating models is a useful way to remove the details from a complex specification or problem, and it allows architects, business analysts and other stakeholders to focus on the important aspects of the problem or solution. Distilling the rules and decisions of an organization or initiative into a single model allows them to be connected into a network (mathematical graph) which immediately enables the way they interact to be visualized. It also provides a mechanism for locating gaps or missing Decisions, Inputs, Business Knowledge and authoritative references.

Enterprise Architect has a number of tools that can be used to find gaps or missing elements in a model. The result demonstrates the power of modeling, namely to find issues or deficiencies before they are baked into a solution and the users of an implemented solution discover them. The rectification can be costly and time consuming and the potential damage to the brand or loss of trust with customers could be commercially damaging or in safety critical domains irretrievable.

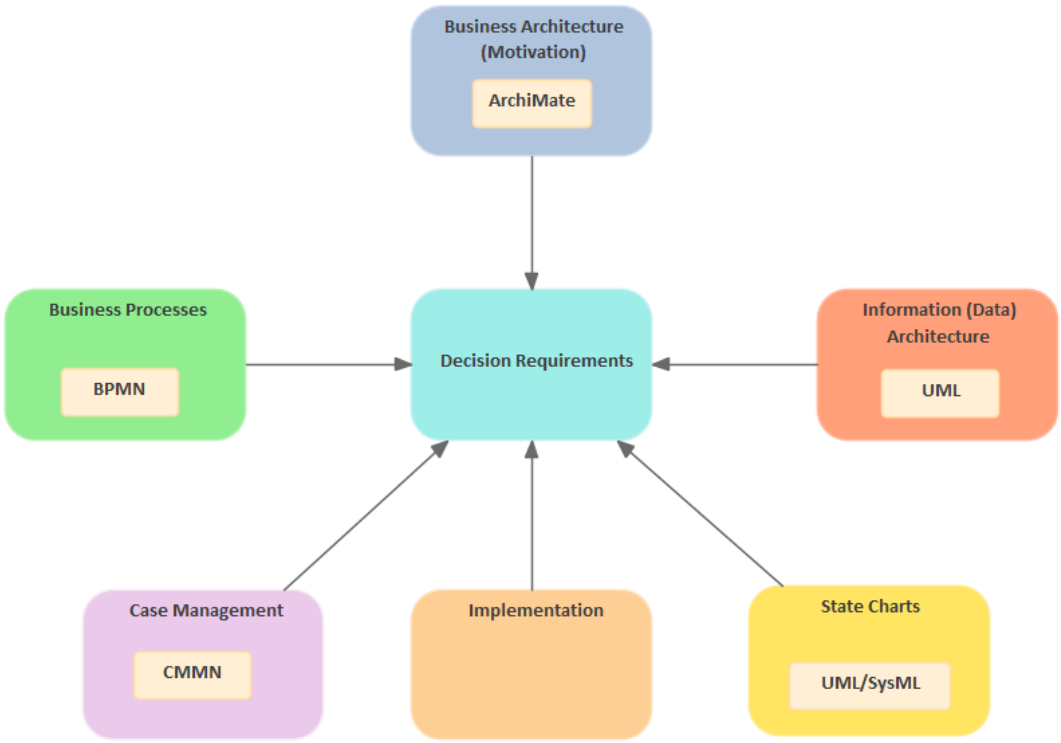


This illustration shows how the Relationship Matrix can be used to quickly locate gaps in the decision graph by visualizing decisions without any Data Inputs. The matrix can be used to find gaps or duplications between any other related elements such as Business Knowledge Models without corresponding Knowledge Authorities. The Traceability window shows another way of visualizing the Decisions and their relationships to other elements such as Requirements. Both windows allow you to locate the elements in the Browser window and any diagrams that contain the objects.



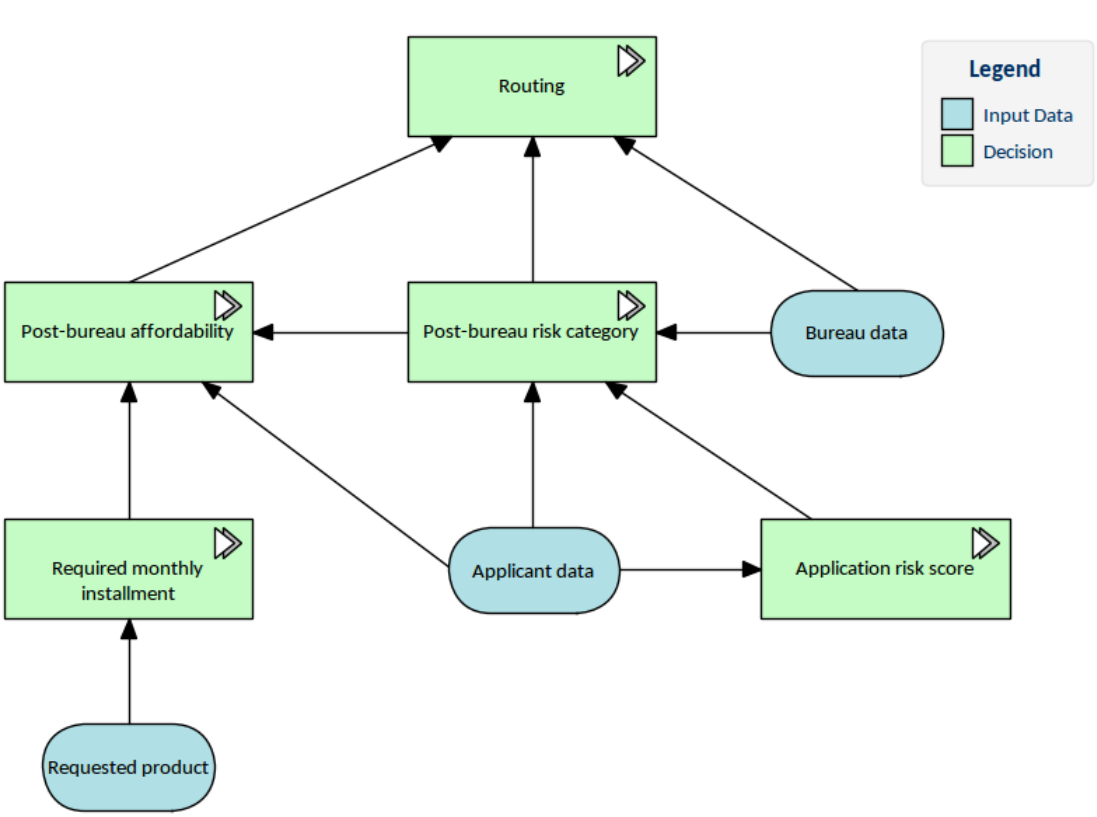
Articulate with other Models

The real power of Enterprise Architect is its ability to relate models from a wide range of disciplines and thus integrate strategy, ideas, knowledge, applications and technology. The tool is uniquely able to do this because of its great depth and breadth of features, ranging from Strategy Maps down to low-level engineering diagrams such as Executable StateMachines and code diagrams. These same multi-dimensional features apply to Decision Modeling, but the power comes from integrating all the models into a Model of Models. For example, Strategy can be linked to decisions including Drivers and Goals, and onwards down to the Implementation code that will drive the runtime engine that drives the decision; all can be modeled in Enterprise Architect.



Visualize Inputs and Analytics

A common reason why decisions are misunderstood or why the outputs from a decision engine or manual or automated process are often a surprise is because they have not been tested before the implementation. An important part of this process is the ability to model and analyze the inputs to a decision remembering that the decision could be modeled as a hierarchy with decisions that act as inputs to other decisions down to any level. Enterprise Architect allows these inputs to be visualized and provides a mechanism for creating any number of Data sets that will assist in testing canonical examples and boundary and unusual data points.



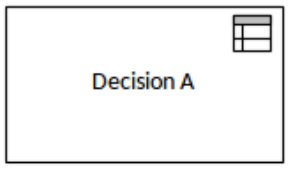
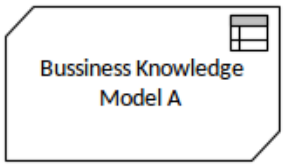
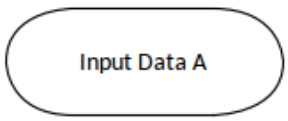
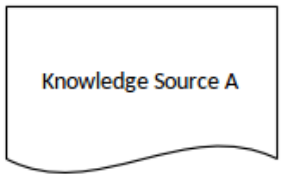
Synopsis of the Notation

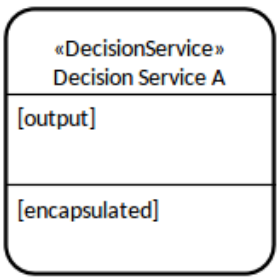
A Decision Requirements diagram is a graphical representation of a graph that uses the relationships between elements and connectors in the diagram to describe a Decision Model. The connectors are referred to as Requirements in the specification but in the diagram are represented as lines that are used to connect the element into a graph. The tables here list the elements and connectors that are available from the DRD Toolbox pages and that can be placed onto a diagram to create the Decision Requirement level of the model. Effectively, this is the level that would typically be created and maintained by the business analyst, architect or other business personnel.

Elements of a Decision Model

The elements are 2-dimensional shapes that are used to model the various parts of the decision requirements diagram they are available from the DRD toolbox. The elements are the nodes that make up the digraph (directed graph) and are connected by relationships (referred to as requirements) in the form of lines. The elements are used to define a range of concepts from Decisions themselves to Knowledge Sources.

Component	Description
Decision	A *decision* denotes the act of determining

	an output from a number of inputs, using decision logic that can reference one or more business knowledge models. A decision can be represented in a number of ways, by a Decision Table, Invocation, Context, or Literal Expression.
Business Knowledge Model 	A *business knowledge model* denotes a function encapsulating business knowledge, e.g. as business rules, a decision table, or an analytic model. It serves as a reusable component that could be stored in a library and could be included in any number of Decision Models.
Input Data 	An *input data* element denotes information used as an input by one or more decisions. When enclosed within a knowledge model, it denotes the parameters to the knowledge model. The information defined in the input data element can be structured.
Knowledge Source 	A *knowledge source* denotes an authority for a business knowledge model or decision. The information is external to the decision model and their effect is a continuum from being mandating

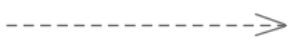
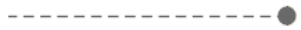
	(regulation or law), controlling (policy), guiding (best practice), influencing (recommendation). The information in the knowledge source vary widely in form from a document, web page, printed material, video or audio content.
Decision Service Expanded 	A *decision service (expanded)* denotes a set of reusable decisions and serves as an invocable element, connected with a knowledge requirement connector to other elements with invocation logic. It provides a mechanism for packaging parts of a decision model into a component or service based architecture and provides an interface that specifies the required information inputs and the resulting outputs. Using this expanded form a modeler can show the details of the service including its encapsulated and output compartments.
Decision Service Collapsed	A *decision service (collapsed)* denotes a set of reusable decisions and serves as an invocable element, connected with a knowledge requirement connector to other elements with invocation logic. It provides a mechanism for packaging parts of a decision model into a component or service based architecture

	and provides an interface that specifies the required information inputs and the resulting outputs. Using this collapsed form a modeler can hide the details of the service including its encapsulated and output compartments.

Requirements (Relationships) of a Decision Model

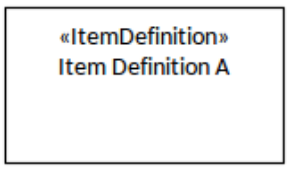
The Decision Model and Notation standard defines three relationships that can be used to connect components in a Decision Requirements diagram. The relationships are directed, and their application creates a digraph (directed graph) connecting the various components of the model. There are two line types used (solid line, dashed line) and three connector end markers (a closed arrowhead, an open arrowhead and a filled circle), which are described in this table.

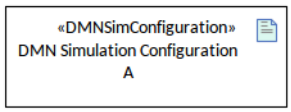
Information Requirement	An *information requirement* denotes input data or a decision output being used as one of the inputs of a decision. It therefore specifies the data that is consumed and processed by the decision to determine the outputs. It is represented as a solid line with a solid arrowhead.

Knowledge Requirement 	A *knowledge requirement* denotes the invocation of a business knowledge model. This indicates that a decision, a decision service or another business knowledge model invokes a business knowledge model to receive its outputs. It is this mechanism that effectively allows a business knowledge model to be reused in different models and contexts.
Authority Requirement 	An *authority requirement* denotes the dependence of a Decision Requirements Diagram (DRD) element on another DRD element that acts as a source of guidance or knowledge. This is a useful mechanism for providing an explanation of the origin of the information that is being used to make the determination of the element making the reference.

Artifacts

There are two types of artifacts listed here, the artifacts that are part of the specification and an additional two elements; one of which is part of the specification but not seen in the notation list and another which is an Enterprise Architect artifact used to visually configure the simulation of a model.

Text Annotation 	A *text annotation* provides a mechanism for a modeler to add explanatory text or a comment to the model. These annotations do not have any modeling semantics but provide informal information related to the entire diagram or to a specific set of elements. They can float in a diagram or be attached by an Association to one or more model elements to indicate their applicability.
Association 	An association is a connector that links a text annotation to one or more components in the decision model diagram. It does not have any other semantics beyond this.
Item Definition 	An *item definition* is used to model the structure and the range of values of input data and the outcome of decisions, using a type language such as FEEL or XML Schema. An Item Definition is used to define the structure of the input data and optionally, to restrict the range of allowable values of the data. Item Definitions can range from a simple single type through to a complex structured type. The core properties of an

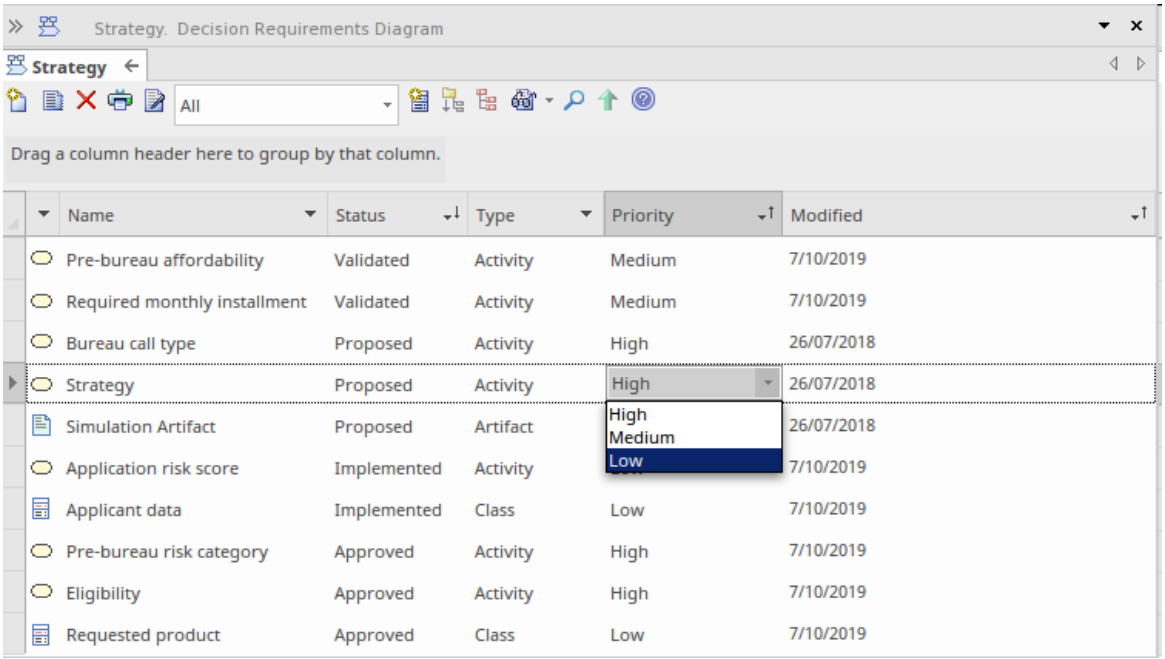
	Item Definition element are accessed via the DMN Expression window.
Simulation Configuration 	The DMN simulation configuration is an artifact that is used to specify the configuration of a simulation. It allows the Package that contains the decisions to be specified and the values set that are used for input data. It is not part of the DMN specification but a part of Enterprise Architect's modeling environment, and is used to provide a visual mechanism to configure simulations. The artifact must be placed on a diagram that contains decisions that are to be simulated.

The Decision Requirements Diagram

In the *Getting Started* topic we introduced the basic concepts of working with diagrams in Enterprise Architect and we explained how to create a Decision Requirements diagram (DRD), add a number of elements to the diagram and connect the elements with relationships. The DMN is a visual language and the diagram is the main way that you will work with decision models, including creating them, updating them and simulating them. It is important, however, to understand that a diagram is just a view of the underlying repository and that it is used to communicate with a given audience, and the author might deliberately elect not to display all the related elements. For example, when communicating with the business stakeholders it might be important to show the Knowledge Sources that are related to the Business Knowledge Models, and so a diagram could be created showing these relationships. When communicating with a technical audience the author might consider these to be of limited value and create another diagram not showing the Knowledge Sources. The two diagrams will share many elements, and updating properties of elements on one diagram will also mean they are updated on all other diagrams where they exist. In the next section we will look at a number of facilities for working with diagrams.

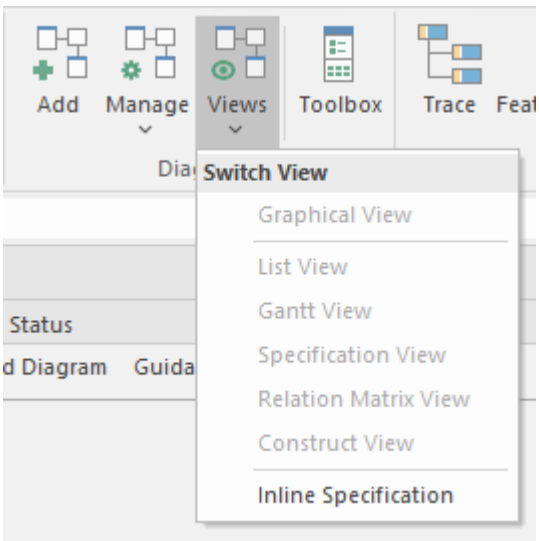
Alternative Views

Even though the diagram is the most common way to work with a set of elements the tool provides a range of other ways to view the elements in a Package or diagram. These provide great flexibility, are useful in particular circumstances and are also helpful for people who find it easier to work with other modes of presentation such as lists.



These options are available from the diagram's context (right-click) menu and also from this ribbon option:

Ribbon: Design > Diagram > Views



One of the most useful of these is the *List View,* which presents the diagram elements in a list and allows their properties to be visualized and edited as if in a spreadsheet. This is particularly useful for properties such as Status, Version and Author. A number of properties are displayed by default but other properties, including Tagged Values, can be selected and added by using the header context (right-click) menu.

Another unique way to visualize the elements in a diagram or Package is the *Specification View*. This presents as both a word processor and a spreadsheet view providing a comforting experience for those business or technical modelers who are more familiar with these types of office automation tools.

There are two other views that are useful when resources have been assigned to the elements -the Gantt and Construct views. The Gantt chart shows a useful view that can be centered around a resource or an object.

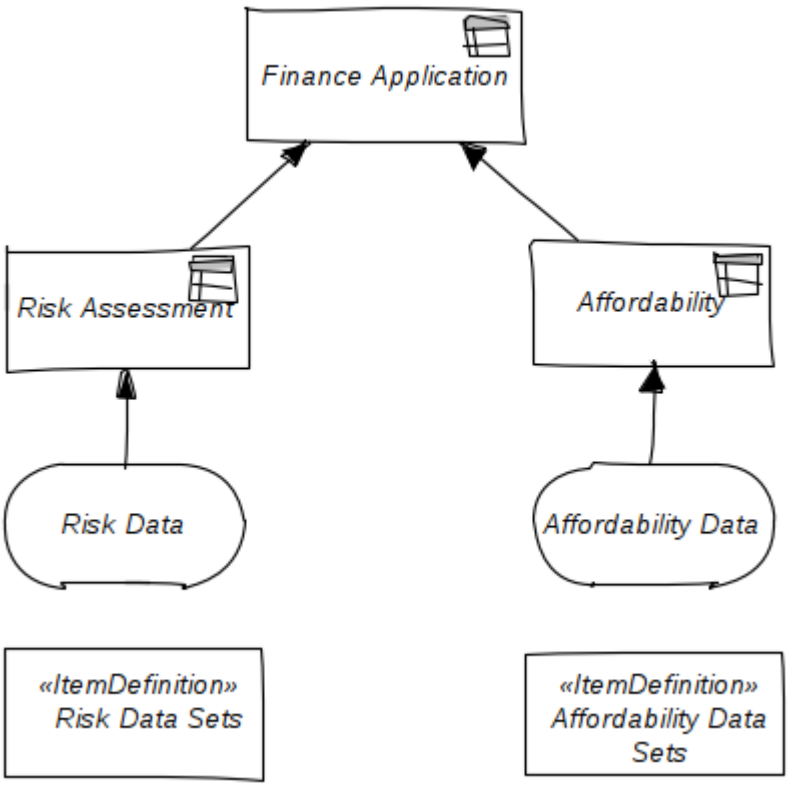
Modes and Styles

Diagrams can be viewed in a number of different modes that provide a useful ways of presenting the diagram information in various business contexts. There are three modes or styles that can be applied to a diagram.

- *Hand Drawn Mode* - applies an effect that the elements

have been hand drawn

- *Whiteboard Mode* - applies the hand-drawn effect and also changes the elements fill color to white as though the diagram had been drawn on a whiteboard
- *Custom Style* - allows great flexibility with label positions and rotations, transparency and more to create visually compelling diagrams

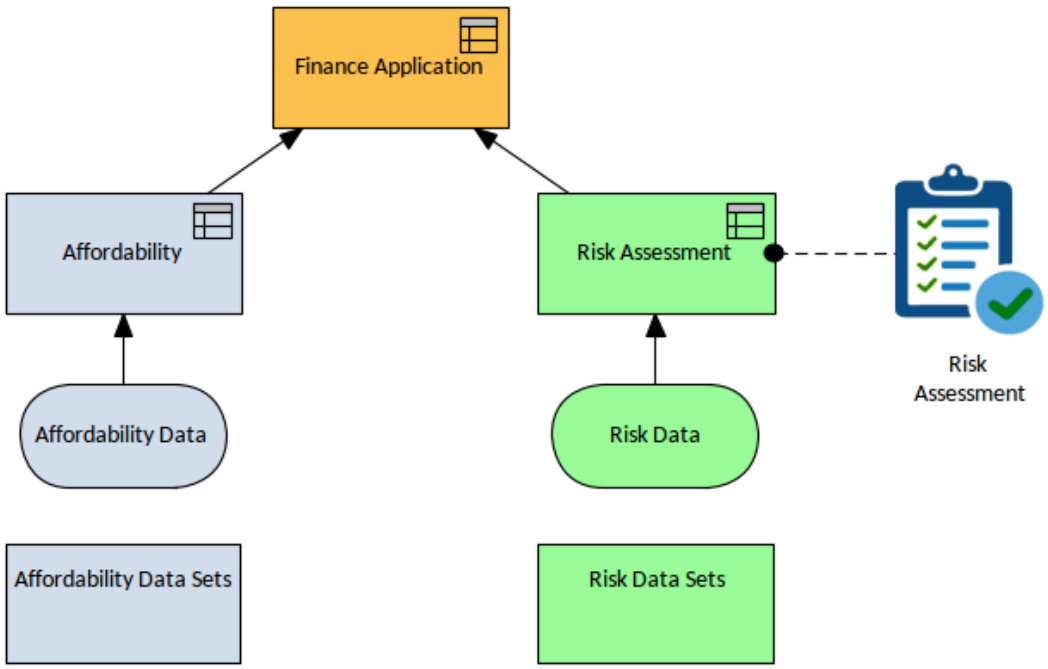


Layout Ribbon

Diagrams have the most impact when they are well structured and neatly laid out, and Enterprise Architect provides a large number of tools for creating compelling and highly polished diagrams that will be perfect for presentations to executive level, business and technical

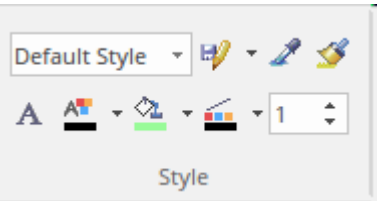
stakeholders alike. We have already looked at diagram themes, modes and styles, and now we will look at how to change the appearance of elements on the diagram.

Many stakeholders prefer images in diagrams, and adding some image content can make the diagram more compelling and approachable for those stakeholders. An example of the use of an image would be to replace the Knowledge Source element with an image representing the organization or source of the authority. In this diagram we see the use of an icon representing a risk assessment; colors have been used to show the separate inputs to the high level decision, and some details such as stereotypes have been suppressed in the diagram.

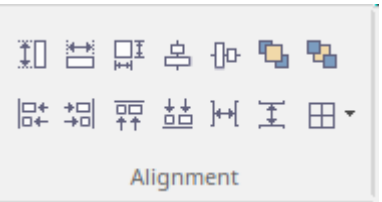


In addition to diagram themes and styles, Enterprise Architect also allows styles to be set for each individual element. These Styles include fill, border and font colors, and line and border thickness. These settings and others are

available from the style panel of the Layout ribbon and the element icons on the right hand side of a diagram object.

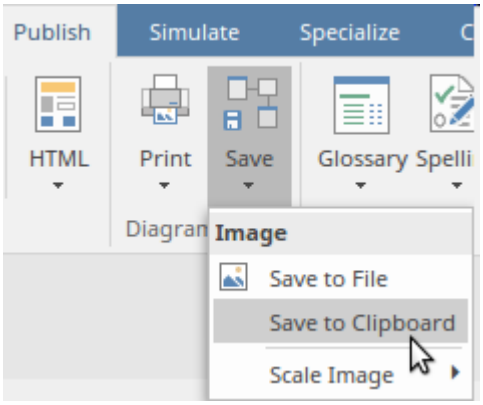


There is also a useful set of alignment tools that can be applied to a selected group of elements to set uniform heights and widths, align edges and centers horizontally and vertically, and more.



Exporting

The model content in Enterprise Architect, including diagrams, can be shared with other users and tools, allowing the model content to be visualized in other contexts. Diagrams can be simply copied to the clipboard and then pasted into any other application such as a slide presentation, word processor document, web page or email. This option is available from the *Diagram Image* panel of the Publish ribbon.



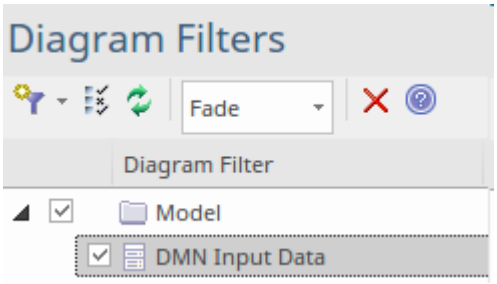
There is also a useful diagram report that can be used to export all diagrams contained in a Package, including sub-Packages. The diagram can be exported to a variety of formats, including gif, bmp, wmf and emf. Elements in the Decision Model can also be exported to a CSV file for opening in a spreadsheet application.

When high quality documentation is required Enterprise Architect can be used to generate first class publications using a number of pre-defined or custom templates. The template engine provides a large number of configuration points and allows an organization's corporate styles to be imported, including cover pages with images that will make the final document appealing to strategic and business stakeholders.

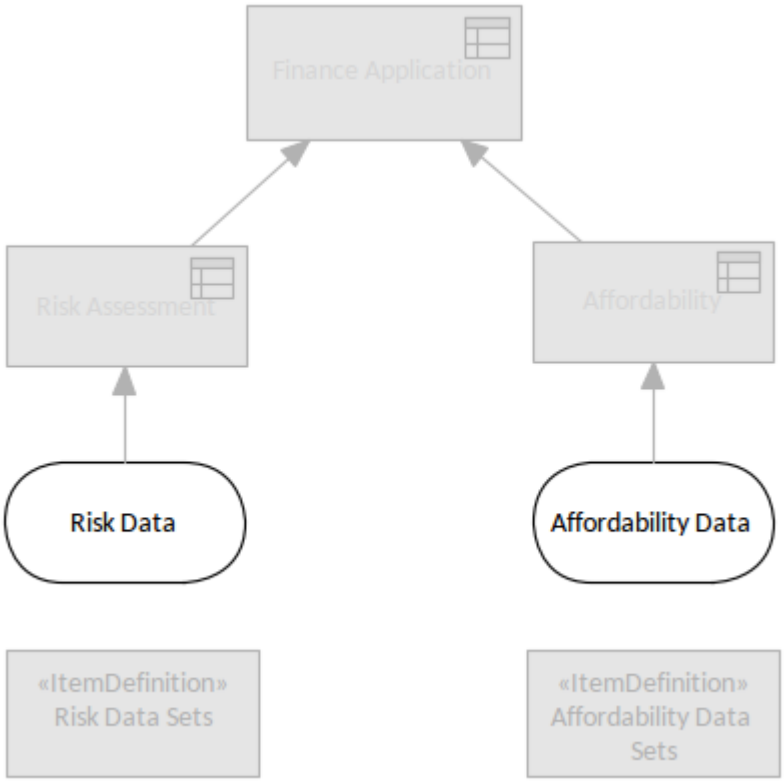
Filters

Diagram Filters are a handy focusing device where elements in the diagram can be either hidden or represented in a faded or altered style to raise the prominence of the remaining elements that are the focus of the diagram. The filters can be

toggled and are a useful device for presentations or publications so the audience can visualize the elements that the author wants to draw their attention to. The filters can be based on a search and can be dynamically applied to any diagram with the same effect.



This diagram shows the effect of applying this filter; notice the 'effect' chosen is *Fade* meaning that diagram elements that are not of type Input Data will be faded in the diagram and just the Input Data elements will remain at full strength. Other effects can be chosen providing useful options for visualizing the elements depending on how the diagram is being viewed and the audience present.

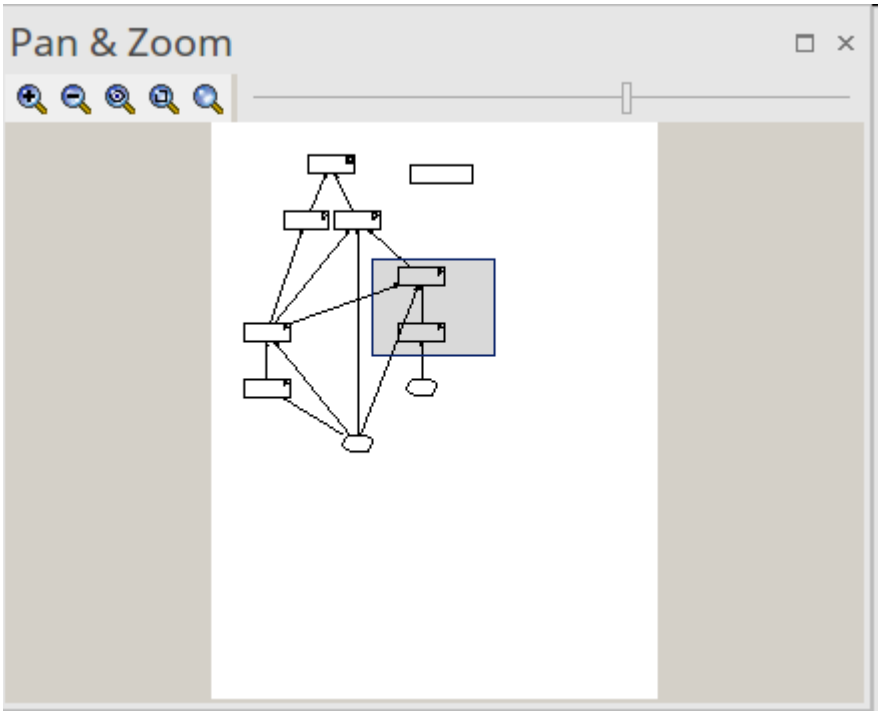


For more information see the Visual Filters Help topic.

Pan and Zoom

The Pan and Zoom facility is one of the tools that can be used to navigate around a large decision diagram. Often the resolution of a diagram must be reduced to ensure it is wholly visible, but by using the Pan and Zoom window you can leave the diagram at a readable resolution and pan around to areas of interest, zooming in when necessary. Even when you are fortunate enough to be using a large monitor, you will often want to change the scale at which you are viewing the diagram and then pan around to find the section or element of interest in the diagram, zooming into

that section to see a more detailed view. The Pan and Zoom window will allow you to do this for any size diagram, with options for panning and zooming that are particularly useful during workshops or focus groups organized to discuss the model with an audience who might not be familiar with the diagram.

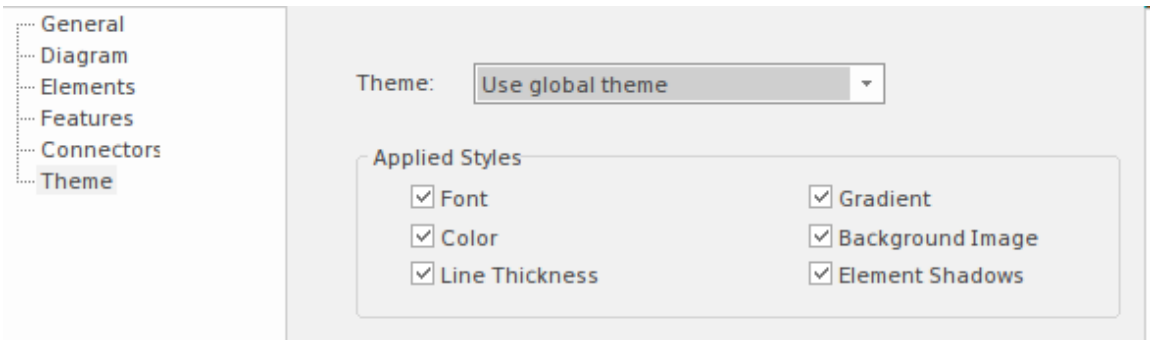


For more information see the Pan and Zoom topic.

Themes Colors and Styles

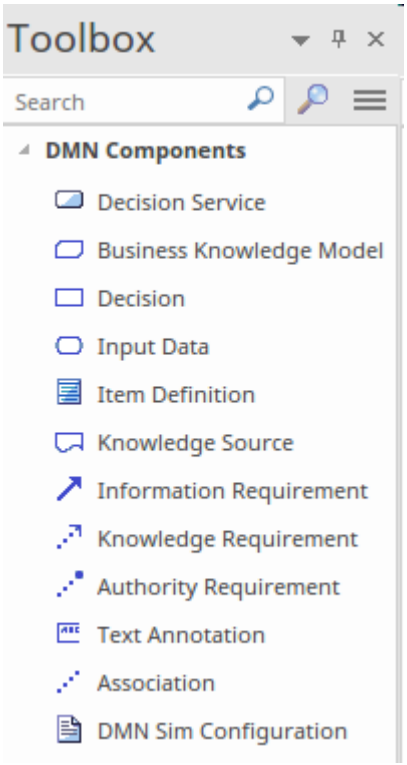
The use of color is an important visual cue and when used carefully and consistently can make diagram more appealing and visually compelling. There is a range of tools available starting with the diagram theme that can be set at a global level but this setting can be overridden for individual diagrams. It is also possible to override the default settings

for an element's appearance including: its size, font, background and line colors the font type can also be set for individual elements. Elements can also be represented by an image either as a default for every diagram where the element appears or just for a specified diagram.



Toolbox

You will recall that, when you create a new Decision diagram or open an existing one, Enterprise Architect will display the diagram in the main canvas and also display the DMN Components Toolbox page, which provides a range of items including elements, relationships (requirements) and artifacts, which can be dragged and dropped onto the diagram.



While the elements and connectors contained in the DMN Components Toolbox page are the items a modeler needs to create Decision Requirements diagrams (DRDs), any number of other elements can be included on the DRD, or any number of elements can be copied to another diagram that can be embellished. This was described in an earlier part of this Guide, where decisions were linked to other strategic elements such as goals and objectives.

Decision Expression Types

The logic level of Decision Modeling and Notation uses expressions that allow decision modelers to construct statements that relate input values to output values using a range of expressions, languages and their predefined functions. The business, engineering or scientific context for each Decision Model is likely to vary widely, and just as the Decision Requirements model will need to differ in these separate contexts, so too will the way the decision logic level is expressed. It is to accommodate these different circumstances that the specification provides four different ways to express the expression logic, and it is left up to the modeler to select the most appropriate one to use. The four value expression types are: *Literal Expressions, Boxed Context, Invocations* and *Decision Tables*. The Decision Table is undoubtedly the most flexible and universally applicable of the expression types, and while they will be discussed here an entire topic in this Guide is devoted to how to work with them.

In the next sections we will provide some guidance on which expression type to use in a given context, but in many ways the order that these types are presented is a reflection of their applicability and complexity. Enterprise Architect supports all four expression types, and adds a range of additional features beyond what is prescribed in the specification that will make a modeler productive and help to ensure that the Decision Models provide great business, engineering or scientific value.

We will also provide some information about the expression languages, including the **F**riendly **E**nough **E**xpression **L**anguage, or FEEL for short.

Value Expression Types

Boxed Context

A boxed context is a collection of context entries, consisting of (name, value) pairs, each with a result value.

The context entries provide a means of decomposing a complex expression into a series of simple expressions, providing intermediate results that can be used in subsequent context entries.

Decision Table

A decision table is a tabular representation of a set of related input and output expressions, organized into rules indicating which output entry applies to a specific set of input entries.

Invocation

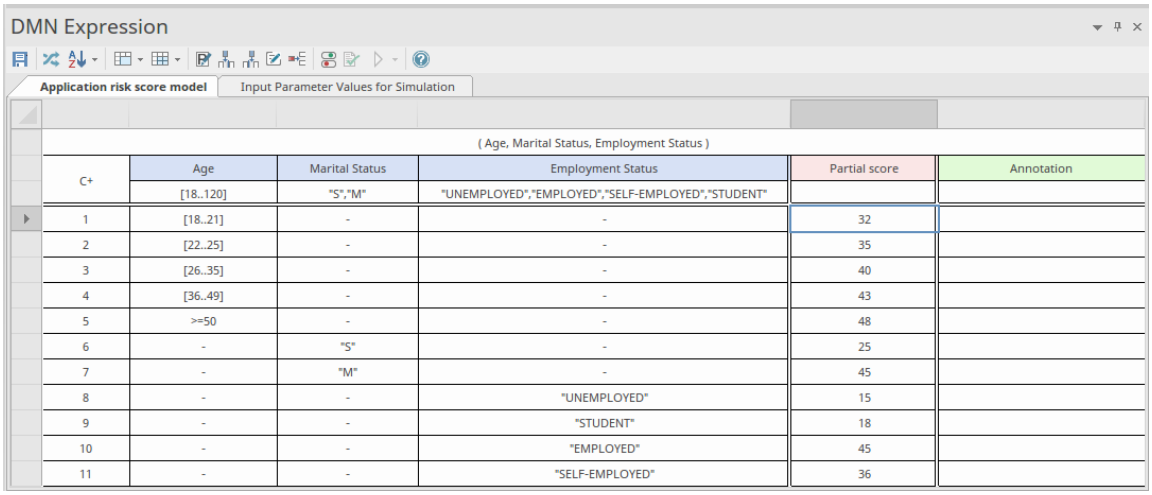
An invocation calls on another model element (a BusinessKnowledgeModel or a Decision Service) to provide a decision result. The invocation defines parameters that are passed into the 'invoked' element, providing context for evaluation of its decision logic. The decision result is then

passed back to the 'invoking' element.

Literal Expression A Literal Expression specifies the decision logic as a textual expression that describes how an output value is derived from its input values. To support simulation and execution, the Literal Expression can use JavaScript functions.

Decision Table

This icon in the top right corner of the Decision element or Business Knowledge (BKM) Model element indicates that it is implemented as a *Decision Table*. Decision Tables are the most commonly used of the expression types because of their tabular format, which is familiar to Business Managers and other non-technical staff who commonly work with spreadsheets and other tabular representations. Fundamentally, the Decision Table is a tabular representation of a set of related input and output expressions, organized into rules. The rules - which can be organized horizontally or vertically - indicate which output entry applies to a specific set of input entries. For example, we might have two rules, one that says that if a customer has a 'Gold' credit rating and spends $20,000 or more a year they will receive a discount of 20%, and a second rule that says if they have a 'Gold' credit rating and spend less than $20,000 they will receive a discount of 15%. When the Decision Table is put into production the input values would be supplied each time a decision is executed and, depending on the values, the customer would receive either a 15% or 20% discount.


DMN Expression

Application risk score model | Input Parameter Values for Simulation

(Age, Marital Status, Employment Status)

C+	Age	Marital Status	Employment Status	Partial score	Annotation
	[18..120]	"S","M"	"UNEMPLOYED","EMPLOYED","SELF-EMPLOYED","STUDENT"		
1	[18..21]	-	-	32	
2	[22..25]	-	-	35	
3	[26..35]	-	-	40	
4	[36..49]	-	-	43	
5	>=50	-	-	48	
6	-	"S"	-	25	
7	-	"M"	-	45	
8	-	-	"UNEMPLOYED"	15	
9	-	-	"STUDENT"	18	
10	-	-	"EMPLOYED"	45	
11	-	-	"SELF-EMPLOYED"	36	

The image shows the key parts of the DMN Expression window for the definition of Decision Tables. Recall that there are three other types of value expression and the window will appear differently for each of them. This list contains the main parts of a Decision Table:

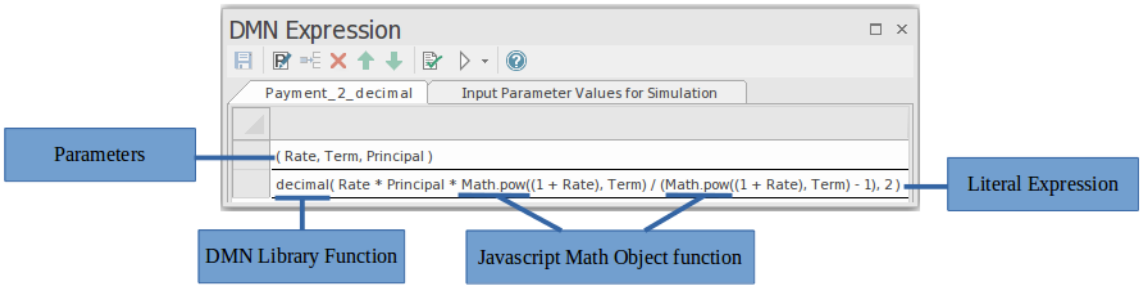
- A list of rules, where each rule contains specific input entries and corresponding output entries
- A list of Input Clauses, defined as expressions that typically involve one or more input values
- A list of Output Clauses, defining the output corresponding to a specific set of inputs
- The Table Hit Policy that specifies how the rules are applied

An Input Clause consists of an expression and an optional list of allowed values. Very often, the expression is simply an unmodified input value; however, it could also be an expression involving more than one input value, or perhaps a conditional statement such as 'Application Risk Score > 100'. The allowable values apply to the *expression result* rather than the input values used.

Each Output Clause consists of an identifier (a name) and, again, an optional list of allowed values for that clause. The Table itself consists of a list of numbered rules, defining a set of input entries and corresponding output entries. The Decision Table should contain all the inputs required to determine an output (and only those inputs). In determining which rules are applied, the expressions defined in the Input Clauses are evaluated for the given inputs and the *expression results* are then used to find rules with matching input entries.

Literal Expression

=X This icon in the top right corner of the Decision or Business Knowledge Model (BKM) element indicates that it is implemented as a *Literal Expression*. A Literal Expression is the simplest form of DMN expression. It is commonly defined as a single-line statement or an if-else conditional block. When an expression becomes more complex, a modeler might choose a Boxed Context in preference to a Literal Expression, or in order to improve the readability the modeler could encapsulate some of the logic as a function in the DMN Library. The Literal Expression is a type of value expression used in both Decision and Business Knowledge Model (BKM) elements.



This image shows the key parts of the DMN Expression window for the definition of Literal Expressions. Recall that there are three other types of value expression and the window will appear differently for each of these.

The Literal Expression is a textual representation of the decision logic. It describes how an output value is derived from its input values, using mathematical and logical operations.

The DMN Expression window presents the Literal Expression as a table, with two key rows:

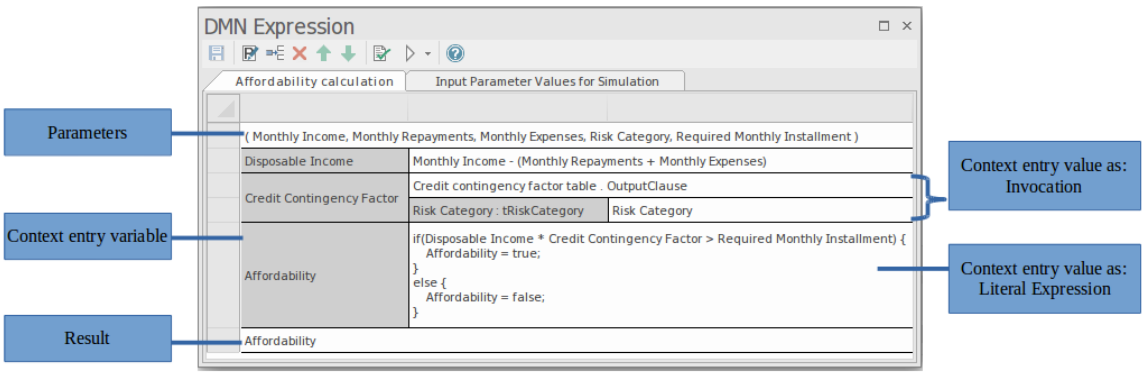
- Parameters: *defines the input parameters used in the expression*
- Literal Expression: *where the formula for the expression is defined - this defines the output of the Decision*

In order to support simulation and execution, the Literal Expression can use JavaScript global functions or JavaScript object functions. Users can also create DMN Library functions for use within the expressions.

Boxed Context

This icon in the top right corner of the *Decision* or *Business Knowledge (BKM) Model* element indicates that it is implemented as Boxed Content. A Boxed expression is typically used in the circumstance that a Literal Expression would result in complex and difficult to understand logic. Like the Decision Table its tabular form makes it easier for business managers and other non-technical stakeholders to understand.

Fundamentally a *Boxed Context* is a collection of context entries, presented in the form of a table, followed by a final result expression. These context entries consist of a variable paired with a value expression, and can be thought of as intermediate results that can be used within the value expression of any subsequent context entry. This allows for complex expressions to be decomposed into a series of simple expressions, with the final result being evaluated in a much simpler form.



This image shows the key parts of the DMN Expression window for the definition of Boxed Context. Recall that there are three other types of value expression, and the

window will appear differently for each of them.

The value expression of a context entry can be either a Literal Expression or an Invocation and can make use of any available inputs, such as parameters (to a BKM element), InputData or decision results, as well as any previously defined context variables. The final result of a Boxed Context is determined by working through each context entry in turn, evaluating the value expression and assigning its result to the variable, then finally evaluating the result expression. The result expression can also make use of any input or local variable, but it must be able to be evaluated to provide a result.

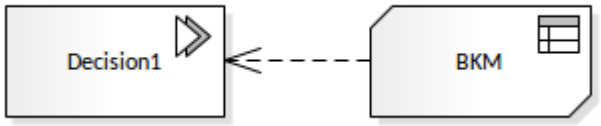
Invocation

This icon in the top right corner of the Decision or Business Knowledge Model (BKM) element indicates that it is implemented as an *Invocation*. The Invocation expression type is the mechanism by which decision expressions can be accessed and reused in different contexts. You will recall in an earlier topic we discussed the Decision Modeling notation and introduced the Business Knowledge Model (BKM) as a method of reusing predefined decision logic. The invocation expression type is the method that is used to access the decision logic in a BKM, providing values that are passed through to a BKM or Decision Service's parameters and in turn receiving the output. The invocation can be applied to both Decision elements and BKM elements, both of which would invoke a Business Knowledge Model or a Decision Service.

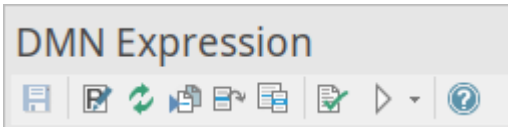
An invocation is a container for the parameter bindings that provide the context for the evaluation of the body of a Business Knowledge Model. There are two common use cases for an Invocation:

- Used to bind Input Data to a Business Knowledge Model
- Used to bind parameters or context entry variables to a Business Knowledge Model

An invocation is a tabular representation of how decision logic that is defined within an invocable element (a Business Knowledge Model or a Decision Service) is invoked by a decision or by another Business Knowledge Model.

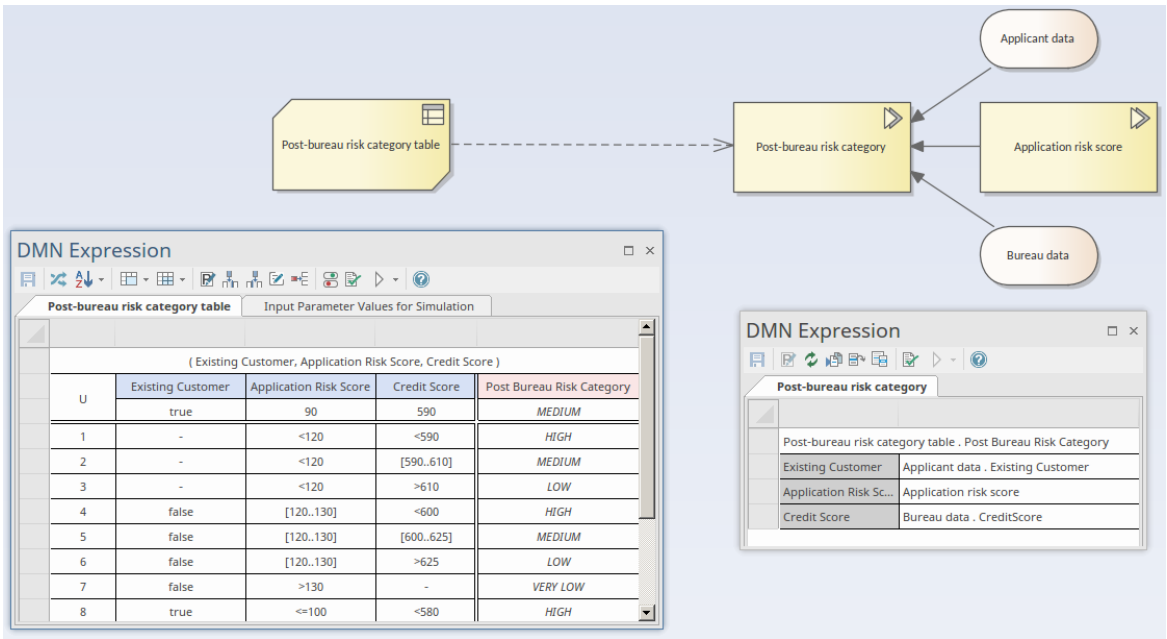


When an Invocation is selected, the layout of features accessible in the DMN Expression window is:



For more details refer to the *Toolbar for Invocation Editor* Help topic.

The parameter bindings of an Invocation provide the context for evaluation of the body of the invocable element.



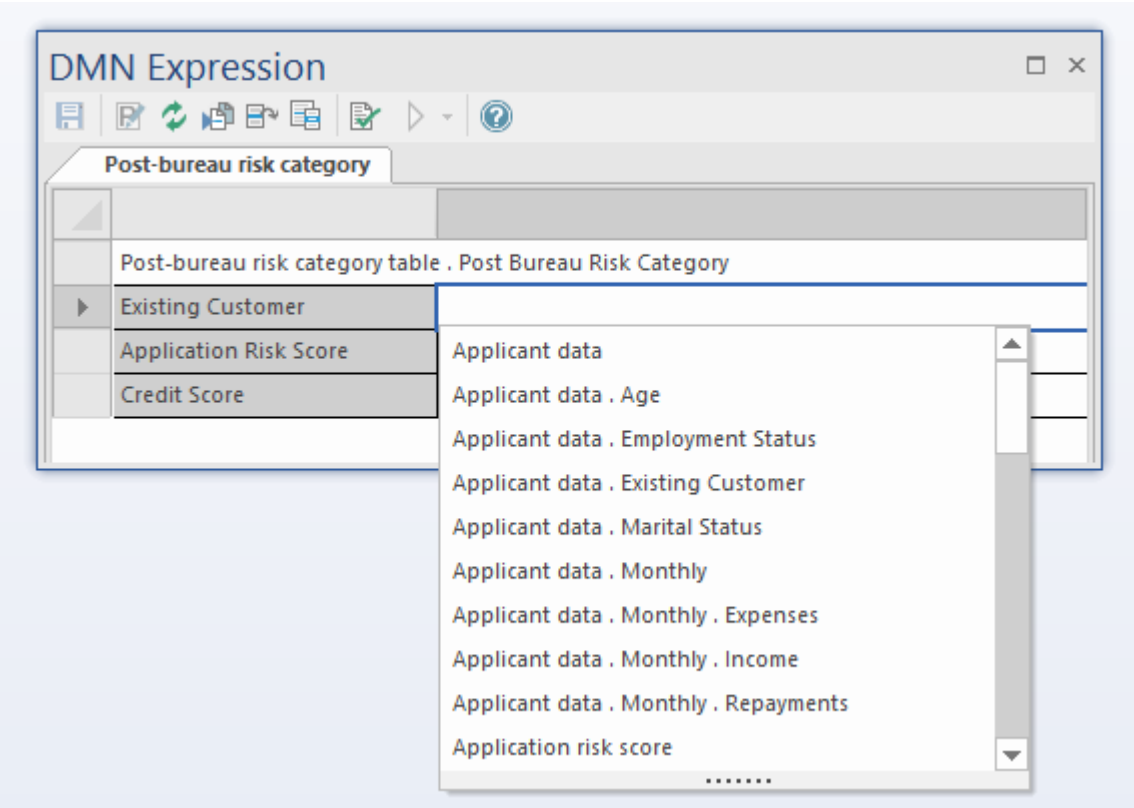
In this example:

- Decision 'Post-bureau risk category' is represented as an invocation connecting to Business Knowledge Model 'Post-bureau risk category table', implemented as a Decision Table
- Decision 'Post-bureau risk category' is the target of three information requirement connectors from two input data

and one decision

- The binding list binds the input values to the Business Knowledge Model's parameters
- The invocation also specified the requested 'OutputClause'; in the case where a Decision Table has multiple Output Clauses defined, the invocation must explicitly request an Output Clause as the result of the expression

Inputs from other Decisions and InputData elements can be set by pressing the Spacebar in the field.



As an Invocation can only invoke one Business Knowledge Model the output is defined by the Business Knowledge Model output.

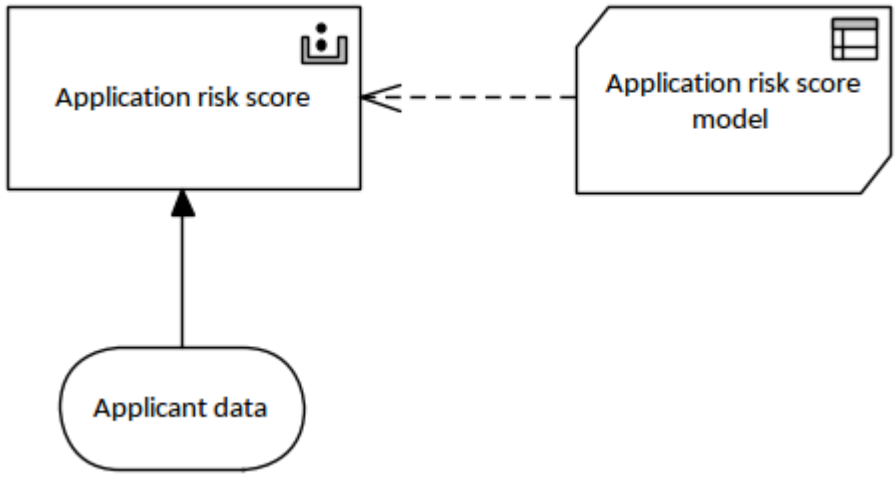
Expression Languages

This is where the rubber hits the road and all of the work that has been done at a level of the Decision Requirements diagram will need to be embellished with the expressions that express the logic of decisions. As with many aspects of design, things start simple but inevitably can become quite complex. It is important at this point in the Decision Model Definition to remember that the purpose of the models is to ensure that the decisions are decomposed into understandable chunks that make sense to the business, engineering or scientific stakeholders. For many applications the logic and the accompanying expressions will be simple enough for most stakeholders to understand. The value expressions that are used in the various expression types are written in the **F**riendly **E**nough **E**xpression **L**anguage or FEEL for short. The next sections will detail the fundamental aspects of the language and how it is supported by Enterprise Architects code editor and Validator.

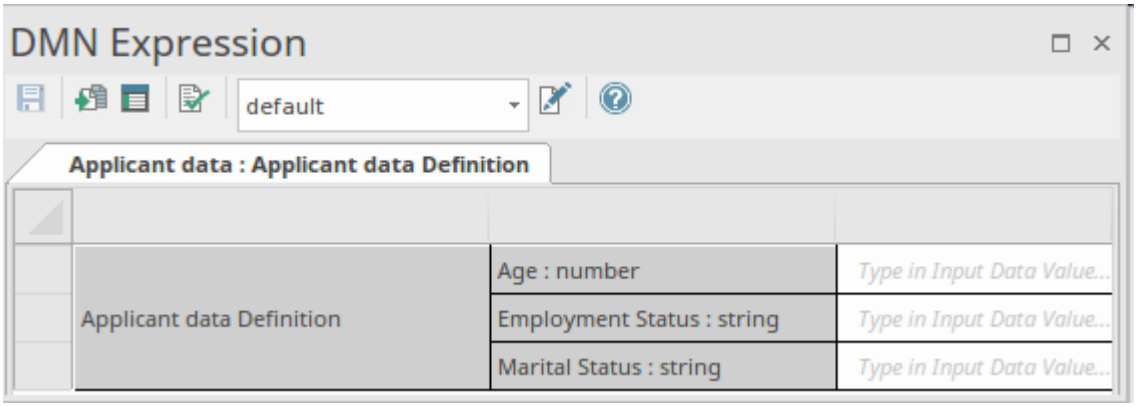
The expressions consist of both literal values (e.g. 20, 'Approved') and variables (Credit Rating, Price) in addition to FEEL language constructs (e.g. not([10..35]), date and time("2022-02-04T07:42:00")).

Expression Editor

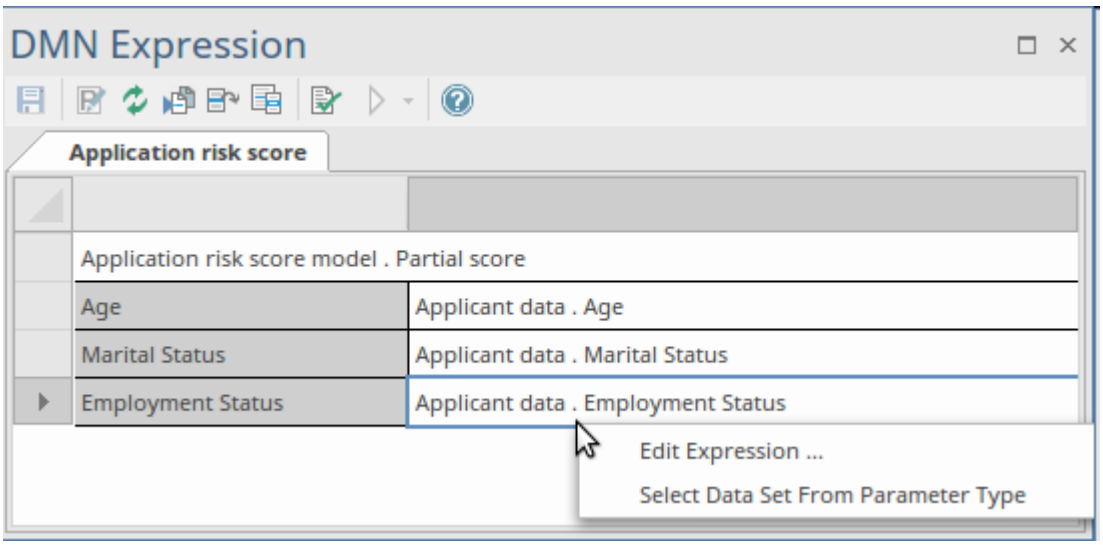
The expression editor is a productivity tool that can be used to create expressions when defining the logic in the DMN Expression window. This diagram shows a simple model with a decision with an expression type of Boxed Context that uses a Business Knowledge Model element. We will look at how we can use the intellinsense otherwise knows as a code completion to



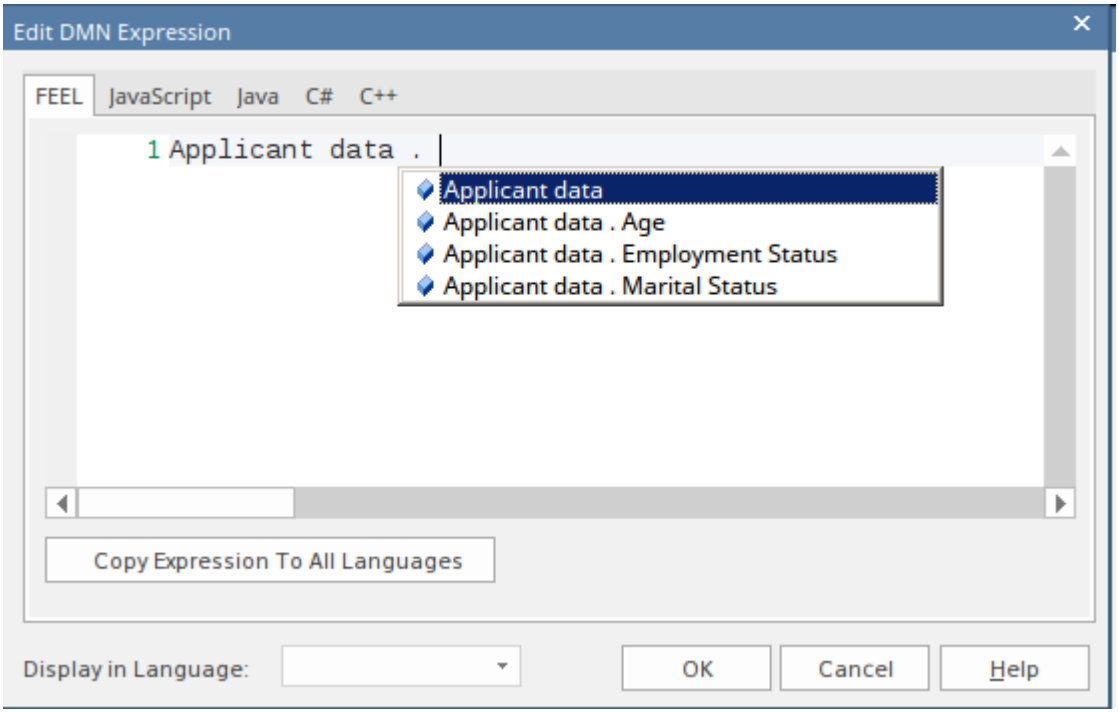
The expression Editor dialog is used for setting expressions in the Boxed content, Invocation and Literal Expression element types. It provides Intelli-sense support for constructing expressions based on the FEEL grammar, as well as the code languages that can be used for the code generation of the model.



This diagram shows the multi-purpose Expression window that is used to define the input data that, as you will see from the diagram, provides input data for the Application Risk Score decision. We will see in the next few steps how Enterprise Architect makes these data items available to the expression editor through intelli-sense. The next illustration shows how to launch the Expression Editor.



With the editor open, the modeler detailing the expression and logic can utilize this productivity tool and reduce the logic errors by accessing the data items with intelli-sense that makes the input data available in a list.


Edit DMN Expression
FEEL
JavaScript
Java
C#
C++
1 Applicant data .
Applicant data
Applicant data . Age
Applicant data . Employment Status
Applicant data . Marital Status
Copy Expression To All Languages
Display in Language:
OK
Cancel
Help

Friendly Enough Expression Language (FEEL)

The Friendly Enough Expression Language (FEEL) was created to support Decision Modeling notation and is defined rigorously in the specification. It is a lightweight language focusing on the creation of expressions with just enough data types expressions and grammar to give it the power to describe decisions. Some more technical readers will question why yet another language is needed when there are existing languages like JavaScript that would be fit for purpose. The answer to this question appears to be that is intended to be simple enough for non-technical people to be able to understand the expressions and contribute to the conversations. Like any language its grammar must be complied with or it will be ill-formed and will not be accepted by the compiler or worse will produce erroneous or unpredictable results.

Enterprise Architect has a facility to generate implementation (programming) code from decision models and expressions defined in FEEL can be automatically converted to a number of different target languages. So there is no misunderstanding let us be clear - FEEL is a language for expressions and is not itself an implementation language.

Data Types

Almost all languages used in computer science have data types, which are intended to help a programmer or analyst specify their intent correctly and ensure the compiler or interpreter is receiving the input in the specified format. A data type is a mechanism to constrain the values that an expression, such as a variable or a function, can take. The data type prescribes the operations that can be performed on the data, the meaning of the data, and the way values of that type can be stored. FEEL has four data types as specified in this table.

FEEL Data Types

Data Type	Description
Number	Number is based Decimal 128 format specified in the on IEEE 754-2008 standard. So they can be integers or decimals. e.g. *123*, *2089, 0.005*
String	Strings are defined as a double-quoted sequence of characters' e.g. "Gold", "Approved", "Alert Sequence".
Boolean	Boolean is defined as the logical pair *True* or *False e.g. Flase*
Date, Time and Duration	There are Date, Time, Date Time, Days

	and Time and Years and Months duration. These include Calendar, Date and Durations. e.g. date("2022-04-06"), date and time("2022-04-06T07:42:00")

Grammar Expressions

Expressions are used to define the logic and contain names and FEEL expressions that define operators and parameters and return values. There are four categories of grammar rules: Arithmetic, Comparison, Interval and Additional. We will now look at these in detail.

Arithmetic

The *arithmetic* expression are some of the most commonly used of the grammar rules and simply apply the arithmetic operators that we would all be familiar with including the exponent (x**n) to raise a number x to the nth power.

Addition (+)	75 + 2.75 (evaluates to 77.25)
Subtraction (-)	100 - 24.5 (evaluates to 75.5)

Multiplication (*)	50 * 5 (evaluates to 250)
Division (/)	144 / 6 (evaluates to 24)
Exponentiation (**)	2**6 (evaluates to 64)
Unary Minus	-200 (evaluates to -200)

Comparison

The *comparison* operators are useful when we want to make decision on degree. There are always two operands involved in the expression with the operator in between in the form x operator y.

Name	Example
Equal (=)	24 = 8*3 (evaluates to *True*)
Greater than (>)	24 > 2**5 (evaluates to *False*)
Greater than or equal to (>=)	

	21 >= 7*3 (evaluates to *True*)
Less than (<)	16 < 25 (evaluates to *True*)
Less than or equal to (<=)	0.25 >= 1/4 (evaluates to *True*)
Not equal to (!=)	25 != 8**3 (evaluates to *True*)

Interval

The interval operators are useful with ranges of numbers that allow the upper and lower bounds to be included or excluded from the range. These are very useful operators but caution must be exercised to ensure that a bracket, round or square, is not misplaced which would inadvertently change the meaning. The validation engine is useful in this situation and can detect unintended gaps or overlaps in expressions.

Inclusive - Inclusive	2 in [2..5] (evaluates to *True)*
Inclusive - Exclusive	5 in [2..5) (evaluates to *False)*
Exclusive - Inclusive	5 in (2..5] (evaluates to *True)*
Exclusive - Exclusive	2 in (2..5) (evaluates to *True)*

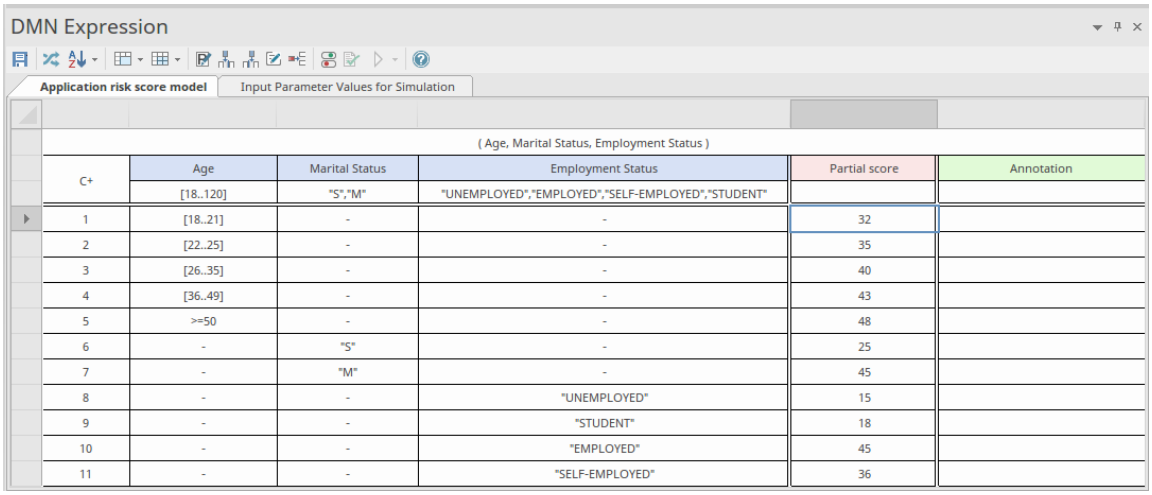
Additional

The additional operators are the logical conjunction, disjunction and Negation operators which are widely understand in formal logic as OR, AND, NOT.

Conjunction	(3*4 = 12) and (3*5 = 15) (evaluates to True)
Disjunction	(3*4 = 12) or (3*5 = 17) (evaluates to True)
Negation	not(3*5=15) (evaluates to False)

Decision Tables Explained

Decision Tables are one of the most commonly used Expressions and offer great flexibility with the definition of rules allowing for logic expression that are logically 'OR-ed' which is not possible in a number of the other expression types.

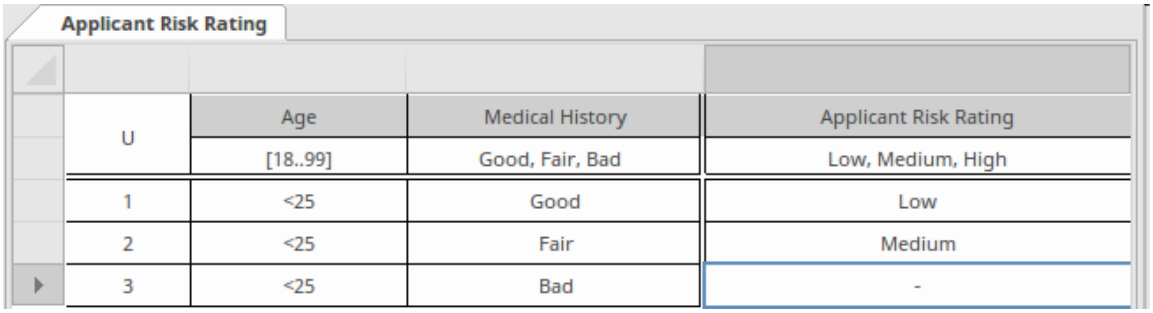


(Age, Marital Status, Employment Status)

C+	Age	Marital Status	Employment Status	Partial score	Annotation
	[18..120]	"S","M"	"UNEMPLOYED","EMPLOYED","SELF-EMPLOYED","STUDENT"		
1	[18..21]	-	-	32	
2	[22..25]	-	-	35	
3	[26..35]	-	-	40	
4	[36..49]	-	-	43	
5	>=50	-	-	48	
6	-	"S"	-	25	
7	-	"M"	-	45	
8	-	-	"UNEMPLOYED"	15	
9	-	-	"STUDENT"	18	
10	-	-	"EMPLOYED"	45	
11	-	-	"SELF-EMPLOYED"	36	

This diagram demonstrates the anatomy of the Decision Table, the parts of which are discussed in the subsequent sections. A Decision Table is essentially a set of rules that will be evaluated when the Decision is being executed. Each rule contains any number of inputs, and when the logic is executed and a rule is selected the outputs defined against that rule will be returned. The input expressions in a given rule are combined as a logical 'AND', so every input expression in a rule must evaluate to True before the rule is selected.

In this example there are two input columns - Age and Medical History. For the output of Applicant Risk Factor = Medium, the Applicants Age <25 AND their Medical History must be Fair.



Applicant Risk Rating

U	Age	Medical History	Applicant Risk Rating
	[18..99]	Good, Fair, Bad	Low, Medium, High
1	<25	Good	Low
2	<25	Fair	Medium
3	<25	Bad	-

Hit Policy

Each Decision Table must have a Hit Policy that specifies how the rules defines in the table will be evaluated. The default is 'U', signifying 'Unique', which means that rules must not overlap and one and only one rule will be selected. There are a number of Hit Policies that return a single row (Unique, Any, Priority, First) and a number that return multiple rows (Output Order, Rule Order, Collect). Hit Policies are explained in greater detail in the later topic *Hit Policies*.

List of Rules

A Decision Table is essentially a set of rules that define inputs and outputs. The rules can be orientated horizontally or vertically and can contain any number of Input expressions and one or more Output expressions.

List of Input Clauses

Each Input Clause consists of an expression defined in each rule of the table. The expressions generally comprise one or more input values, e.g. 'Age>25'.

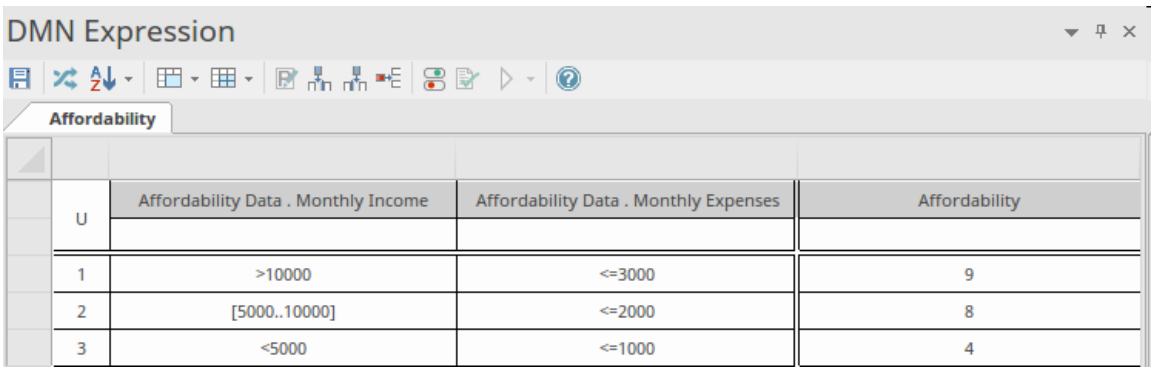
List of Output Clauses

Each Output Clause consists of an expression that can be a simple element or comprising one or more input values, e.g. 'High, Medium, Low'.

The Decision Table toolbar contains a number of facilities for working with the tables, including adding rows and columns, merging cells, and validating the table for syntactic and logical correctness. For more details of the Decision Table toolbar see the Toolbar for Decision Table Editor Help topic.

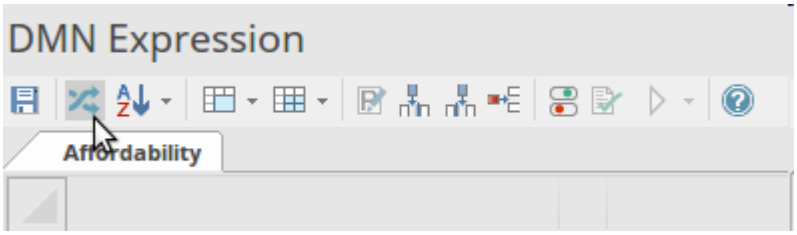
Table Orientation

The specification of the decision table allows the rules to be listed either horizontally or vertically to accommodate different user's preferences for the orientation of the table. This flexibility is simply a question of presentation and the different orientations do not in any way affect the logic or semantics of the rules or the inputs and outputs of the decision table. So a table with horizontally listed rules is logically equivalent to a table with rules listed vertically. Enterprise Architect has a helpful facility to change the orientation of the table. This is very useful, particularly when demonstrating the rules to an individual or to a group of stakeholders during a workshop where one or more of the stakeholders might find it easier to view the rules in a particular orientation.

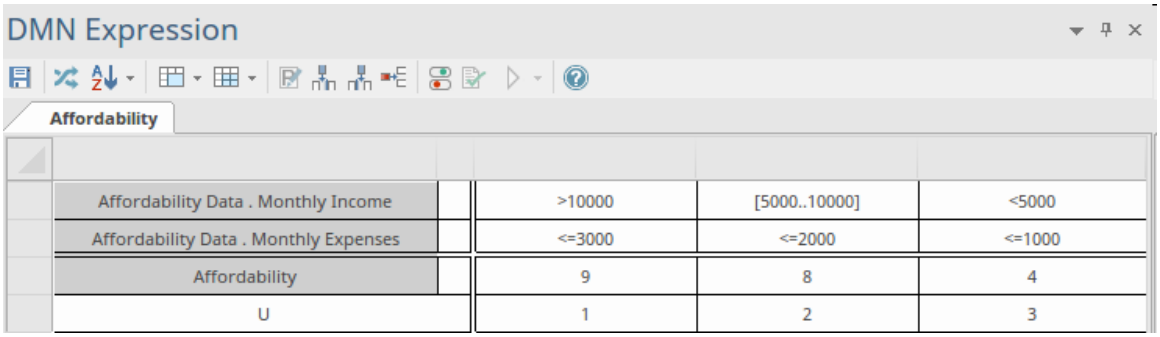


U	Affordability Data . Monthly Income	Affordability Data . Monthly Expenses	Affordability
1	>10000	<=3000	9
2	[5000..10000]	<=2000	8
3	<5000	<=1000	4

The orientation can be changed by simply selecting the 'Rotate decision table' icon on the DMN Expression window toolbar.



This is a toggle icon and selecting it again will simple revert the table to its original orientation. As suggested earlier, if you are working with colleagues who are unfamiliar with DMN or logic rules in general and who seem to be struggling with understanding the decision table, it is worth toggling the orientation and seeing if that helps them.



Affordability Data . Monthly Income	>10000	[5000..10000]	<5000
Affordability Data . Monthly Expenses	<=3000	<=2000	<=1000
Affordability	9	8	4
U	1	2	3

Allowable Value Fields

Input expressions are written in the cells of a Decision Table and are usually quite simple, being either:

- Values, such as 'Gold' or
- Tests, such as 'Age < 25'

The values or variables in these expressions must be provided through Input Data, which should be visible in the Decision Requirement diagram. The universe of available or possible input values or ranges will typically have been discussed at the definition level, and these form an all important reference for the Decision Table definition. For example, the decision for a loan application might rely on an input of *Customer Level,* which could be defined in the business as an enumeration of values - *'Gold, Silver, Bronze'* - and could also depend on a simple *Age* value defined as '*18 < Age < 65*'. It is important when defining the rules and their logic within the table to include these constraints, as it will make the table easier to understand for humans and easier to validate for the tool. Without these being defined a modeler might assume the table is complete, but the validator could possibly prescribe otherwise.

Irrespective of how a modeler defines the expressions, the input values should be exclusive and complete to ensure the table is defined correctly and will produce the expected outputs at run time (after it is implemented and used in practice).

- Exclusive - means that the input values are disjoint; that

is, they do not overlap with each other - no overlaps

- Complete - means all relevant input values defined in the domain have been captured - no gaps

These input constraints or restrictions can be added to the header just below both the input and output columns, in a table with rules oriented horizontally.

Finance Application

A	Risk Assessment	Affordability
	Low, Moderate, High	[0..10]
1	*Low*	>8
2	*Moderate*	[5..8]
3	*High*	<5

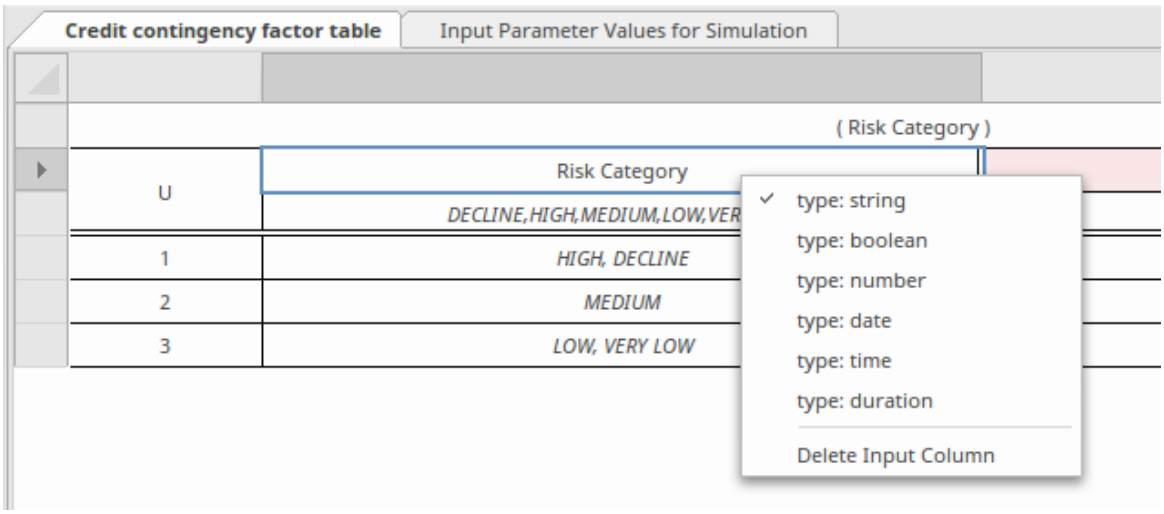
This screenshot shows that a modeler has defined the allowable values for the input columns for the Finance Application Decision Table. The risk assessment input is only allowed to have three values, namely High, Moderate and Low. A value such a Minimal, if listed in the table, would result in a validation error. We will look at validation extensively in a subsequent section of the Guide, but this is an example output from the validation facility in the case where *'Minimal'* is erroneously used, as we have discussed.

```
Running Finance Application Validations ...
       Validating Decision 'Finance Application' ...
               Warning : DecisionTable "Finance Application" Input Violation:
                         Input Value is not allowed for "Rule[7].Risk Assessment": "Minimal"
Finance Application Results: (0) error(s), (1)warning(s)
```

Data Types for Input Output Clauses

To ensure the model is well-formed and the logic is correctly specified and for the simulations to work it is critical to set the data type for all Input and Output Clauses. Range, gap and overlap validations are supported for clauses of type 'number', but validation cannot be performed if the type has not been specified. Code Generation for typed languages such as C++, C# and Java requires that the data types are specified. When the data type is specified as 'string', there is no need to enclose each string literal within quotes. String values are displayed using italic font if the type has been declared.

To set the data type, right-click on the Input Clause or Output Clause and select the required type from the list.



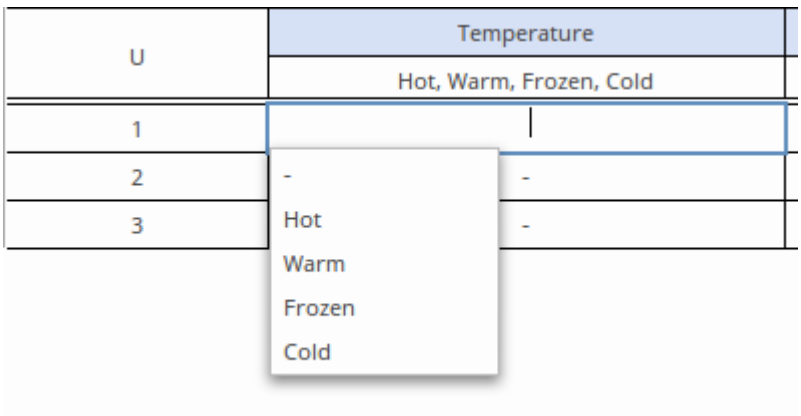
Rules and Inputs and Outputs

Decision Table rules are defined by specifying input entries and corresponding output entries within the cells of a table row. For 'number' data types, input entries can be specified as a single value, or as a number range, such as '<10', '>100' or '(2..8]'. (When defining number ranges, the use of round brackets indicates that the bounding number is NOT included, use of square brackets indicates the bounding number is included.) Output entries should specify a single value per cell.

Additional rules can be appended to the list of rules by clicking on the icon in the toolbar. Unwanted rules can be deleted from the table by right-clicking on the rule and selecting the option 'Delete Rule Row' from the pop-up menu.

Existing rules can be copied and pasted within the table by first selecting the rules, (use 'Ctrl+Click' to add/remove from selection), then using the menu options 'Copy Rules to Clipboard' and 'Paste Rules from Clipboard' to perform the copy and paste. The copied rules can then be modified by selecting and editing individual cell entries.

If the 'Allowed Values' field is set for a string or Boolean expression, the Spacebar can be used for selecting a value from the list of allowed values.



Rules can also be sorted within the table, either by:

- Clicking the icon on the toolbar, then choosing to either 'Sort By Input' or 'Sort By Output', or
- Right-clicking on individual rules within the table and selecting the 'Move Rule Up' or 'Move Rule Down' option from the pop-up menu

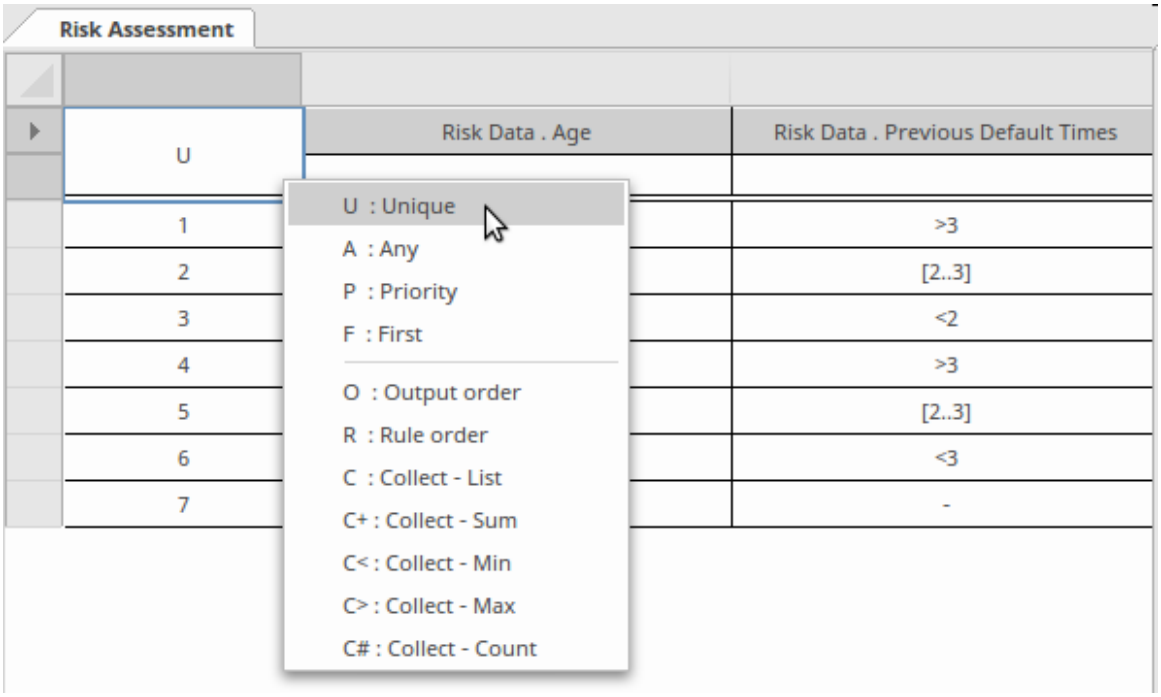
To determine which table rows are selected for output, the *expressions* that are defined by the Input Clauses are evaluated for the given inputs and the *results* of the expressions are then compared against the input entries of the table rows. Where the expression results match the input entries of a table row, that row is selected for output.

The Decision Table's 'Hit Policy' determines how the table's matching rows are then used to produce its output; we will look at what each policy means in the next section.

Hit Policies

The term 'Hit Policies' refers to the way rules are chosen in a Decision Table during the execution of the model. A Decision Table typically contains a number of rules, and in the top left hand corner of the table there is a box that contains a single letter indicating which Hit Policy is applied (and therefore how the rules will be selected). The default is that rules in Decision Tables do not overlap, but if the rules do overlap (meaning that more than one rule could match a given set of input values) the Hit Policy indicator is required in order to recognize the table type and allow the decision logic to be unambiguously understood. Hit Policies can be divided into two groups based on the number of rules that match:

- A single rule is selected (Unique, Any, Priority, First)
- Multiple rules are selected (Output Order, Rule Order, Collect)



Don't be too concerned about what each of these means as we will discuss them shortly and propose some basic recipes for choosing one Hit Policy over another - if in doubt choose the Unique (U) policy as it is the default and is the most commonly used. Enterprise Architect supports all the defined Hit Policies, and the validation facility - which we will learn about in the next section - uses the Hit Policy to determine whether the rules in a Decision Table have gaps or overlaps.

Choosing the correct Hit Policy is critical for the successful specification of the logic level of a Decision Model. As with many other things in modeling, there are Hit Policies that are in practice used more frequently because they turn out to be the best way to express a particular decision, others that are used infrequently and others that a rarely used at all. The Hit Policies include a letter code, name and a description, as shown here.

Single Rule Hit Policies:

- **U**nique (U): no overlap is possible and all rules are disjoint; only a single rule can be matched (this is the default)
- **A**ny (A): there might be overlap, but all the matching rules show equal output entries for each output, so any match can be used
- **P**riority (P): multiple rules can match, with different output entries; this policy returns the matching rule with the highest output priority
- **F**irst (F): multiple (overlapping) rules can match, with different output entries; the first hit by rule order is returned

Multiple Hit Policies:

- **O**utput order (O): returns all hits in decreasing output priority order
- **R**ule order (R): returns all hits in rule order
- **C**ollect (C): returns all hits in arbitrary order; an operator (‘+’, ‘<’, ‘>’, ‘#’) can be added to apply a simple function to the outputs

Collect operators are:

- + (sum): the result of the Decision Table is the sum of all the distinct outputs
- < (min): the result of the Decision Table is the smallest value of all the outputs
- > (max): the result of the Decision Table is the largest value of all the outputs
- # (count): the result of the Decision Table is the number

of distinct outputs

Unique

A table with a *Unique* (U) Hit Policy defines a set of non-overlapping rules, meaning that the rules are mutually exclusive or, in formal set terminology, disjoint. For a given set of inputs, one and only one rule will match, and a single set of applicable outputs will result. It is therefore a 'single hit single output' policy. It is undoubtedly the most commonly seen of all the Hit Policies because of its broad based application to a number of logic contexts, the fact that each rule can be reasoned about independently, and it is easy for business and non-technical stakeholders to understand.

The rule order is in free-variation, meaning that the order of the rules does not affect the outcome of the decision. This has the added benefit of allowing the rules to be ordered in a way that maximizes the overall understanding of the decision logic and also allows rules to be developed and reasoned about as independent entities.

Any

A table with an *Any* (A) Hit Policy defines a set of possibly overlapping rules, with the provision that if they do overlap then the output values are the same. It is therefore a single hit single output policy such that as soon as a rule matches all the input values the result will be returned, as any

subsequent match (which could occur) will result in the same output values. The *Any* policy, like *Unique,* is quite commonly used and its main advantage - and the reason it is used instead of *Unique* - is that it reduces the need to introduce conditions to exclude the condition of rules that would otherwise result in the same outcome. It is therefore always possible to create an equivalent table with a *Unique* Hit Policy, but the table with an *Any* policy will most often be preferred because it is easier to understand and also is not as tedious to create, as there is no need to ensure that every permutation of input values matches one and only one rule.

Offer Category

A	Customer Risk Profile	Customer Years	Order Quantity	Offer Type
	High, Medium, Low			Gold, Silver
1	Low	-	-	Gold
2	-	>5	-	Gold
3	-	-	>=100	Gold
4	Meidum, High	-	<100	Silver

In this example the order of the rules has been defined to facilitate the readability of the table, making the logic more transparent. Notice that the first three rules have the same outcome and have been ordered to make the logic of the table more transparent.

When a table that would otherwise use a *Unique* policy is seen to have a lot of repetitive rules with the same outcomes, consider using an *Any* Hit Policy. One of the benefits of this Hit Policy is that it makes the Decision Tables easier to understand and the logic more transparent.

Priority

A table with a *Priority* (P) Hit Policy defines a set of overlapping rules, meaning that the rules might have differing outputs. It is therefore a single hit/single output policy such that as soon as a rule matches all the input values the result will be returned, as any subsequent match (which could occur) will result in the same output values. The *Priority* policy, like *Any,* is quite commonly used, and its main advantage and the reason it is used instead of *Any* is it supports situations where there is an enumerated list of output values. The order in which the results are listed in the Output column is what determines the rule order and the rule that is selected is the one with the highest priority.

First

A table with a *First* (F) Hit Policy defines a set of possibly overlapping rules with the provision that the order of the rules specifies which rule should be hit. This means that for a given set of inputs the first rule in the table order whose inputs match will be fired and its output will result. It is therefore a single hit single output policy. It is not commonly used and a number of proponents and experienced Decision Modelers prefer not to use it at all, given that it contravenes a best practice rule that the order of rules in a Decision Table should not influence the result. It is, however, quite useful with tables that have a small number of rules and where there are some highly important

and perhaps less frequently occurring conditions that would logically override any more-frequently occurring, less important input conditions.

Merging and Unmerging Cells

Enterprise Architect has a useful feature that allows cells to be merged. It is only possible to merge input cells and not output cells and so these three conditions must be met to be able to merge two or more cells:

- The cells must all be input cells
- The cells must be contiguous
- The cell must have the same value expression

Under these conditions it is possible to merge cells to try and simplify the table and make it easier for non-technical and technical audiences alike to understand. In the decision table in the diagram we can see that there are two groups of cells that meet the conditions and can be merged, namely:

- The two cells where Risk Data.Age < 18 and
- Three cells that have Risk Data.Age defined by the interval [18..55]

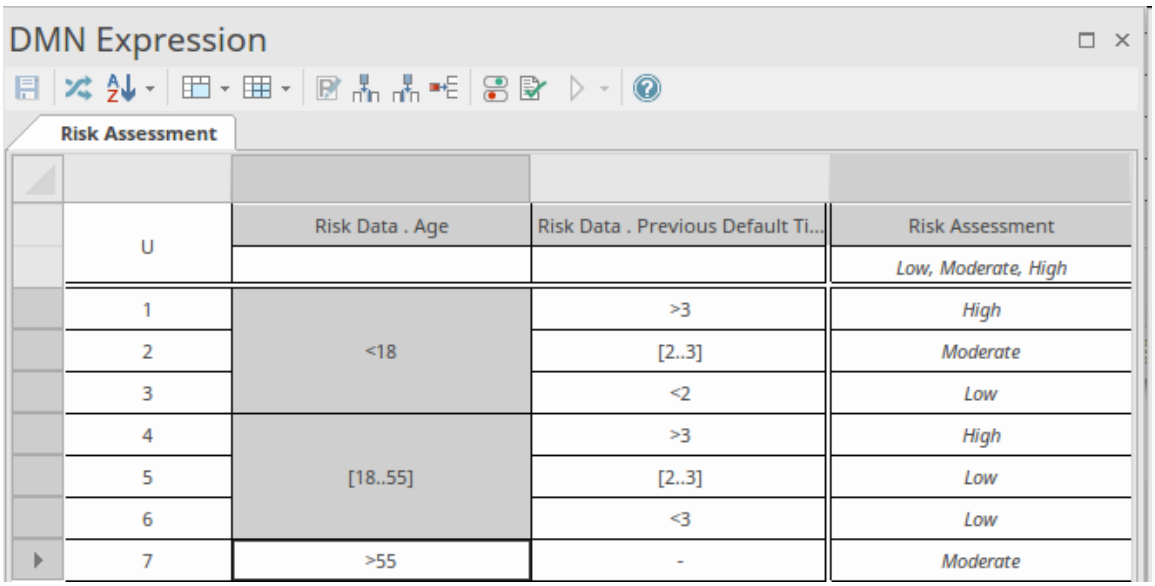
Risk Assessment

U	Risk Data . Age	Risk Data . Previous Default Times	Risk Assessment
			Low, Moderate, High
1	<18	>3	*High*
2	<18	[2..3]	*Moderate*
3	<18	<2	*Low*
4	[18..55]	>3	*High*
5	[18..55]	[2..3]	*Low*
6	[18..55]	<3	*Low*
7	>55	-	*Moderate*

These options are available from the DMN Expression window toolbar. In the case where there is the possibility of more that one merge, there is an option to merge just

specific cells or to merge all cells.

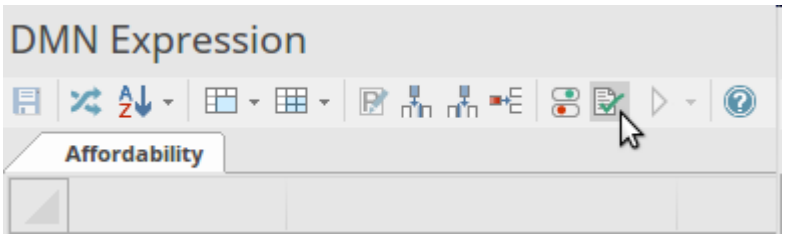
Some Decision Modelers might prefer not to do this, so it becomes a matter of what presentation best suits an audience. Enterprise Architect can assist here because it conveniently provides an un-merge option as well, so if someone has previously merged a number of cells this action can be undone. Clearly this can be toggled to suit different audiences.



U	Risk Data . Age	Risk Data . Previous Default Ti...	Risk Assessment
			Low, Moderate, High
1	<18	>3	*High*
2		[2..3]	*Moderate*
3		<2	*Low*
4	[18..55]	>3	*High*
5		[2..3]	*Low*
6		<3	*Low*
7	>55	-	*Moderate*

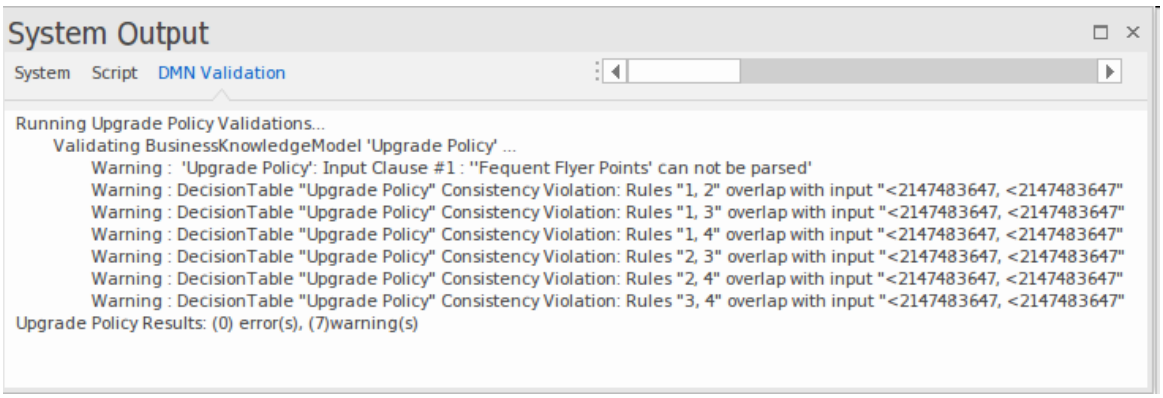
Validating a Decision Model

Apart from being a platform for the collaborative development, management and simulation of Decision Models, Enterprise Architect also has a helpful feature to validate expressions, including Decision Tables, to ensure they are syntactically correct and to find both gaps and overlaps between the rules that have been defined. This feature helps to ensure that when the Decision Models are deployed to a runtime engine or service, the rules will fire correctly and the appropriate result in the form of an output will be provided to the calling application or service. The process of creating rules is often complex, and this validation function will be a welcomed feature for both business and technical stakeholders alike.



The process of finding the business rules and other inputs for a Decision Model can be quite a challenge for a variety of reasons, and when these rules have been collected they will need to be transposed into Decision Rules. They might, for example, be defined in a heterogeneous way and scattered across a range of different sources in different formats. This could present a challenge to the business and so a feature that helps modelers validate and assert that the rules have been entered correctly will assist in ensuring that the Decision Models are well formed and fit for purpose.

Technical staff can also contribute to these models, and the definition of rules - if required - using the collaborative features of Discussions, Chat and Reviews using any device that hosts a browser, such as a smart phone, tablet or Notebook computer. The technical staff can also access the models using the Enterprise Architect client and work together with the business staff to formulate or restructure the rules for optimal understanding, removing redundancy and or missing conditions.



The creation of expressions that accurately define the decisions that are being modeled are like many aspects of technology - both an science and an art. The *science* part is somewhat easier because it can be learnt in a classroom; the *art* part is quite a bit more difficult and is typically learnt from the experience of working with lots of Decision Models. This is evidenced by the fact that when given the same problem a group of Decision Analysts will invariably approach the problem differently and come up with quite different Decision Models; never is this more true than with Hit Policies for Decision Tables, where each analyst will often have a predilection for a particular policy. The models would all be correct, just expressed differently. Enterprise Architect comes to the rescue in these situations with a

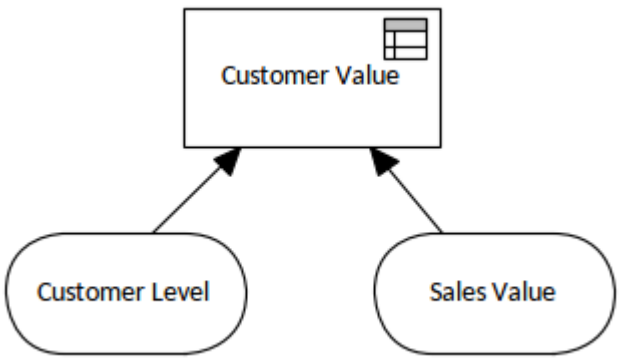
highly compliant implementation of the standard but where the specification is silent, or when an analyst prefers flexibility, the tool provides a number of facilities that will be welcomed by novice and experienced modelers alike.

The DMN standard specifies a number of aspects of the grammar of DMN expressions. including Decision Tables, and it is important that these are complied with; it is equally import that a number of other aspects of the rules are also well formed. The validation of the Decision Tables in Enterprise Architect checks:

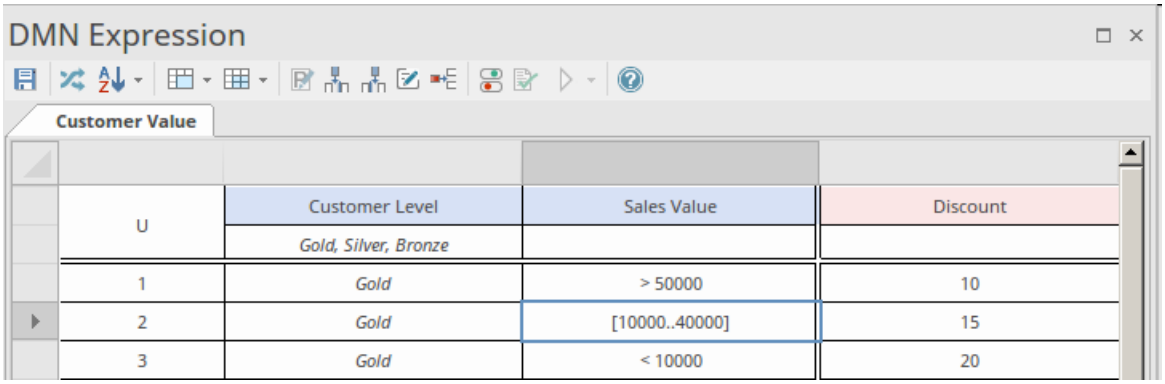
- Syntactical Correctness - ensuring the rules comply with the syntax of the specification and the expression language
- Completeness - ensuring that gaps do not exist between the rules
- Overlaps - ensuring that rules do not overlap

Gaps in Rules

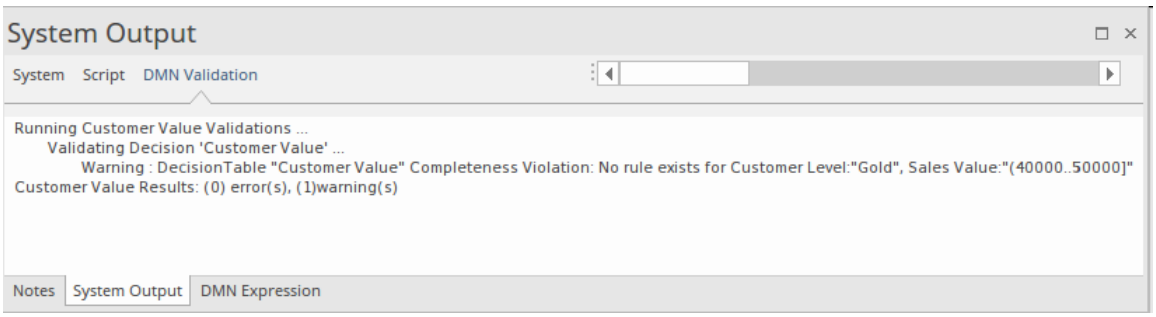
Decision rules must be complete before they are implemented into a production system or service. This is imperative because they must have complete coverage of all possible inputs. There is a catch-all condition in the form of a *Default Output Entry* that can be defined for a table, but many experienced modelers prefer to ensure that all inputs are logically covered. It is also common for less-seasoned modelers to exclude rules for inputs that are not possible within the context of the input domain or universe. A more experienced modeler will know that anything is possible even if the current empirical results suggest otherwise, and will therefore ensure these outlying inputs are covered by the rules. Enterprise Architect's decision table validation can be executed to find any gaps in the rule coverage. This is a profoundly useful facility, as an error in the decision logic can result in unwanted outcomes in an implementation of the rules and even, in some contexts, in catastrophic effects This diagram shows a simple decision model of a single decision with two input data elements; it has been created to exemplify the way the validation can be used.



The second screenshot shows a partially complete decision table with just the *Gold* customers defined in rows. The modeler has inadvertently created rules that contain a gap. This is a simple demonstration and it should be easy for the reader to locate the gap, but in much more complex tables with many more rules and inputs it can be quite challenging to locate the issues.



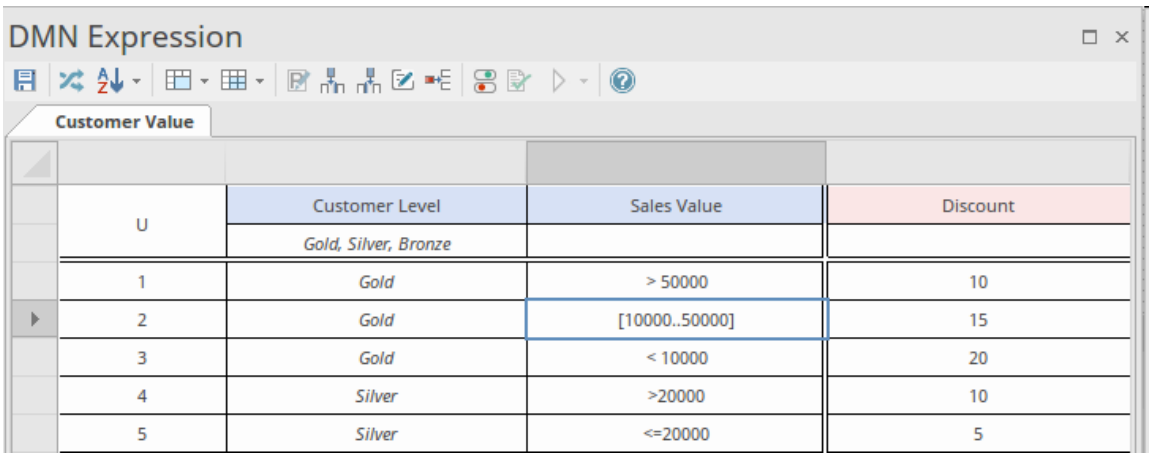
We are able to use Enterprise Architect's built-in validation facility to find any errors with the logic of the rules. In this case the modeler has been remiss and not covered every valid input value. There is not a rule defined for a 'Gold' level customer with a Sales Value of '(40,000 - 50,000]'.



When the validation is run Enterprise Architect will open the output window and display the validation errors, indicating that there is in this case a 'Completeness Violation' and also which rows of the table are implicated. As stated earlier this example is deliberately trivial to demonstrate the feature and it is easy to identify the error by

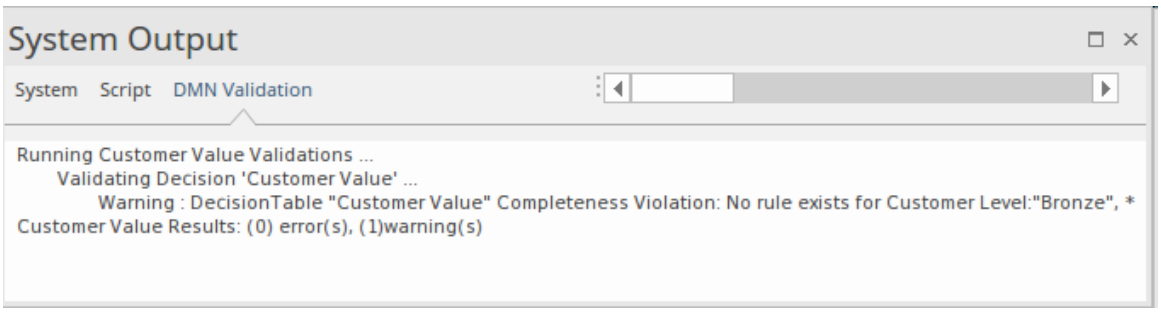
scanning the table rules, but with more complex tables errors can be very difficult to locate.

In this example allowed values have been entered for Customer Level, namely 'Gold, Silver, Bronze'. When defining the rules the decision analyst has forgotten to include rules for the input value of 'Bronze'.

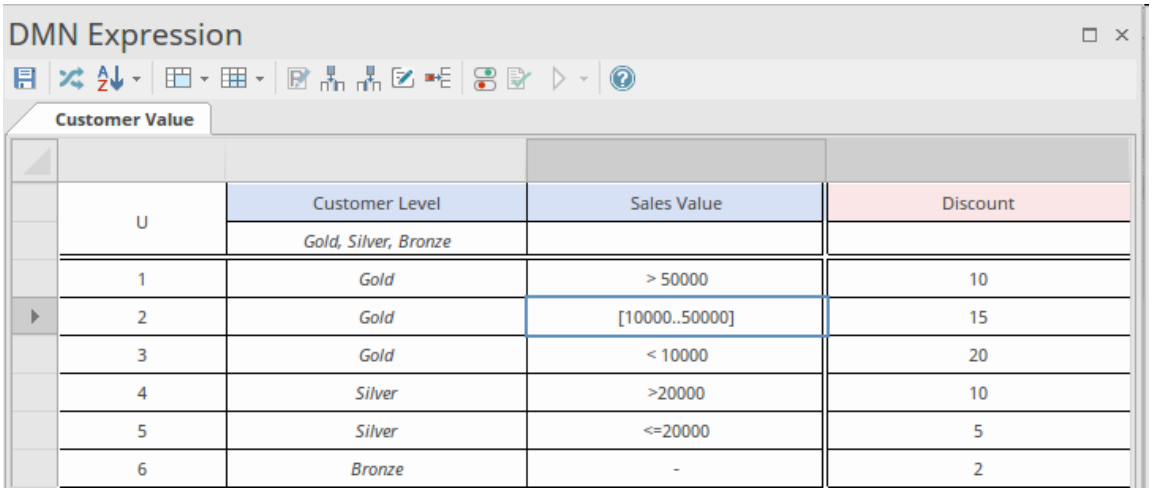


U	Customer Level	Sales Value	Discount
	Gold, Silver, Bronze		
1	*Gold*	> 50000	10
2	*Gold*	[10000..50000]	15
3	*Gold*	< 10000	20
4	*Silver*	>20000	10
5	*Silver*	<=20000	5

When the validation is run the error will be reported as a gap since one of the enumerated values defined for Customer Level has no coverage at all.



The error can be corrected by adding another rule that covers the case where the *Customer Level* is *Bronze*. This would then ensure that the rules are syntactically and logically correct and when the validation was run there would be no errors and no warning generated.



U	Customer Level	Sales Value	Discount
	Gold, Silver, Bronze		
1	*Gold*	> 50000	10
2	*Gold*	[10000..50000]	15
3	*Gold*	< 10000	20
4	*Silver*	>20000	10
5	*Silver*	<=20000	5
6	*Bronze*	-	2

The validation generated to the output window would be:

```
Running Customer Value Validations ...
    Validating Decision 'Customer Value' ...
Customer Value Results: (0) error(s), (0)warning(s)
```

Overlapping Rules

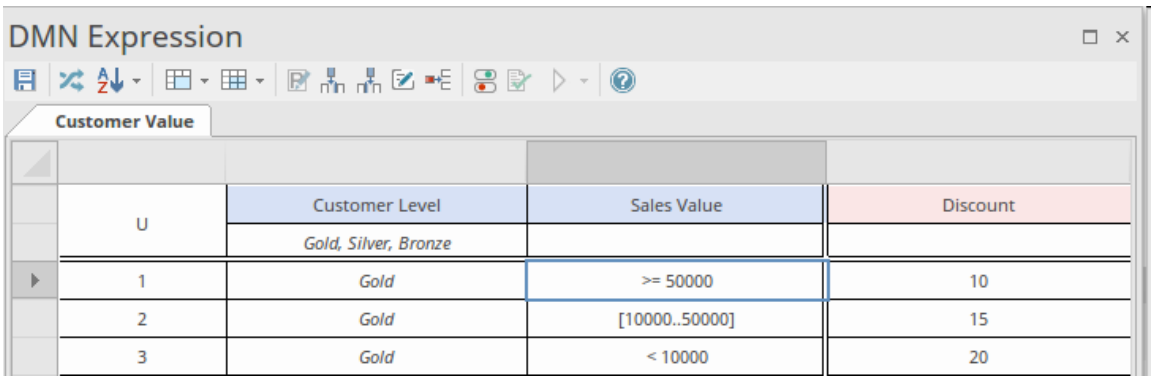
The rules that are defined for a Decision Table with a Hit Policy of 'U' (signifying Unique) must be discrete and not overlap. This is an easy thing to overlook, even for experienced Decision Modelers, and particularly when a table becomes complicated and has a large number of inputs and rules. It is common for overlap errors to be introduced with the use of range expressions in FEEL (Friendly Enough Expression Language) using brackets (round and square) which have different meanings.

Overlapping rules are permitted in Decision Tables with other Hit Policies defined, for example:

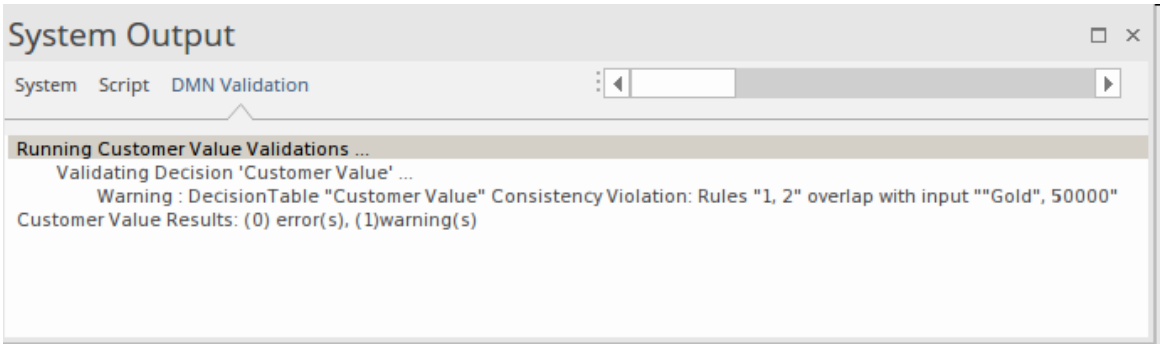
- A table with a Hit Policy of A (Any) can have overlapping rules, as long as all overlapping rules have the same Output Value
- A table with a Hit Policy of P (Priority) can have overlapping rules even when the output values are different
- A table with a Hit Policy of C (Collect) can have overlapping rules even when the output values are different

Using our illustration from the previous section, we will show a simple example of overlapping caused by an error in the use of FEEL expression brackets, as we have just discussed. This screen capture shows an error of two rules overlapping each other, where the problem is a little bit more difficult to identify. The error is introduced because

Rule-#2 includes the use of a square bracket, which can effectively be rewritten as '*Sales Value >= 10,000 and Sales Value <= 50,000*'. The issue arises because Rule-#1 also covers the case where '*Sales Value = 50,000*' so Rule-#1 and Rule-#2 are overlapping.



Once again we can use Enterprise Architect's built-in validation facility to help us identify any violations. In an analogous way to the completeness rules, the validator can find the errors that, if the table were any more complex, would be difficult to find. This illustration shows the violation generated to the System Output window identifying the rules and the values that are in violation.



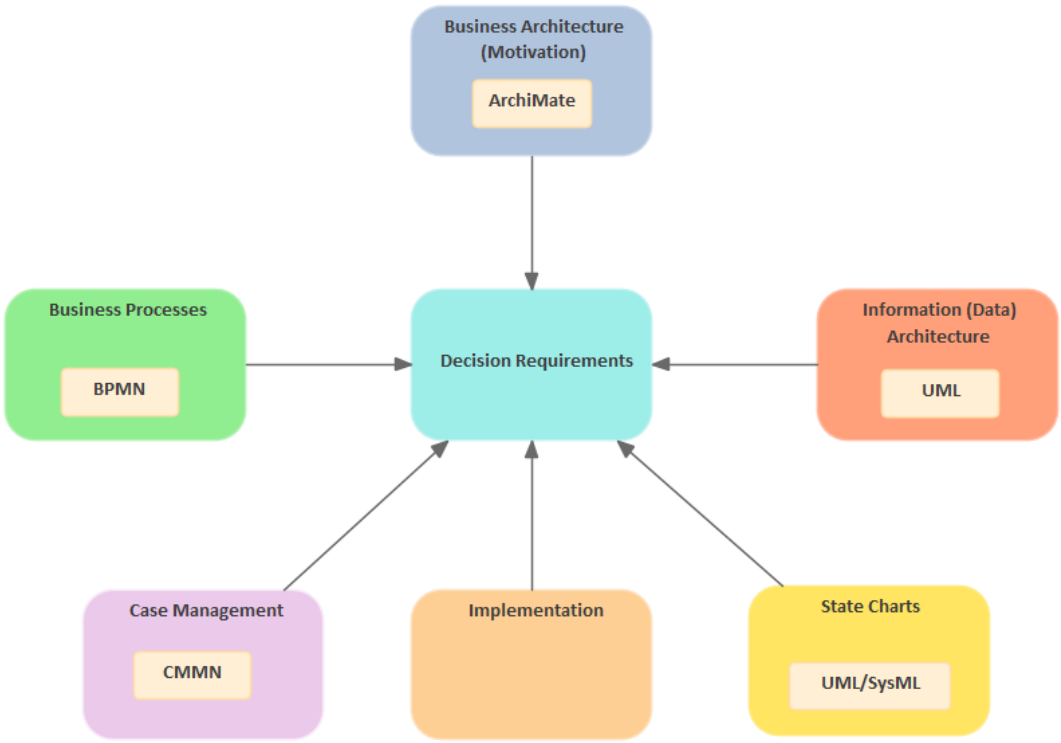
Context for Decision Model and Notation

The Decision Model and Notation (DMN) standard has been created to complement the Business Process Model and Notation (BPMN), used primarily to model Business Process diagrams; the two standards have been designed to work together. BPMN has a specialized Activity called a *Business Rule Task*, which acts as a placeholder for a business rule calculation to be performed by providing inputs and waiting for the outputs provided by the rules engine. This element acts as the departure point into a Decision model, thus allowing complex and often volatile business rules to be defined and managed separately from the process models.

The Decision Model and Notation specification, however, makes it clear that the DMN can stand on its own, independent of BPMN, and that there are a number of other standards and languages that can be used in conjunction with the DMN. This list identifies languages that - by design or by inference - can be used in conjunction with the Decision Model and Notation; new and existing languages will in the future define grammars that interface with the DMN.

- Business Process Model and Notation
- Unified Modeling Language
- Systems Modeling Language
- Case Management Model and Notation

- ArchiMate



Decisions do not exist in isolation, nor are they just the simplification of process models, but rather they are an expression of business intent and often form the fabric of what differentiates an organization from its competitors. The Decision Requirements diagram allows an organization to express how decisions are related, and the Business Process diagram articulates at what points in a set of processes they are invoked. Enterprise Architect is uniquely positioned as an enterprise modeling platform to show how the decisions relate to other enterprise modeling content, including as described in the rest of this topic.

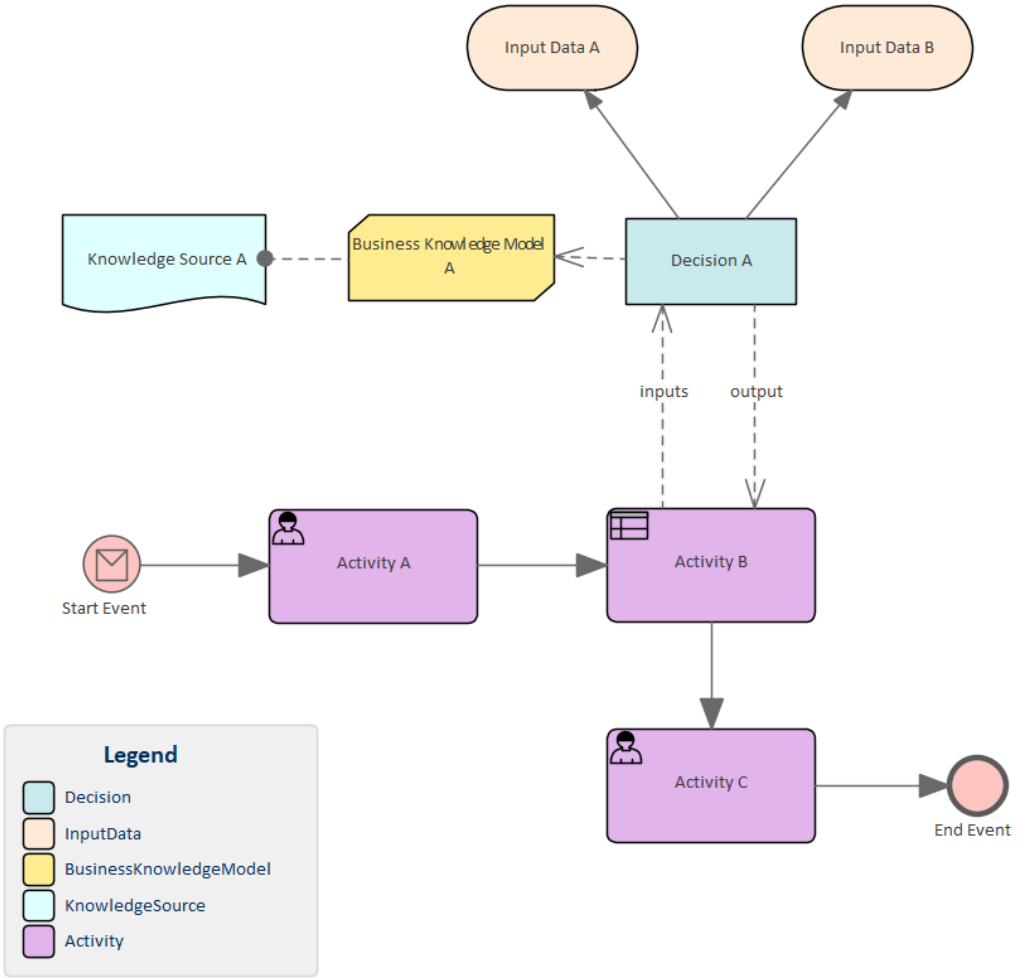
Business Process Model and Notation (BPMN)

This connection between the DMN and BPMN languages

will result in simplified Business Process models and a clear separation between the description of what an organization does and the decisions it makes, ultimately allowing the organization to respond quickly and efficiently to change.

Business Rule Task

The Business Rule Task was added into the later versions of the BPMN specification to allow any Business Rules engine to be called at a specified point in a Business Process, and for the engine to return a result to the process once the request had been processed. It is this mechanism that can be used to integrate Decision Models with Business Processes, thus providing a syntactical mechanism to keep these quite different concerns separate from each other. Enterprise Architect is a rich modeling platform, and it not only allows both types of models to be created but also permits them to be visualized on the same diagram.



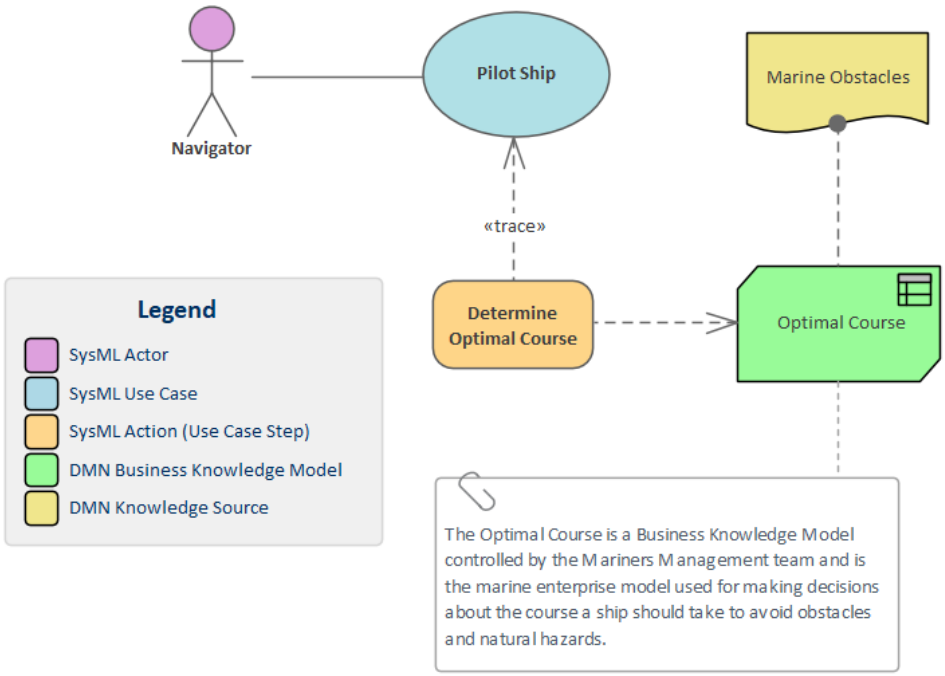
This generic diagram shows how these models can be viewed within the tool; the composite view is simply one option for displaying the two models, and in this view the modeler has only included the highest level Decision and the two contributing Decisions. Full details of the Decision model could be viewed by simply using the Decisions context menu and selecting 'Find | Find in all Diagrams'.

Systems Modeling Language

The Systems Modeling Language (SysML) is used extensively by systems engineers working with a methodology known as Model Based Systems Engineering (MBSE) to describe complex real world systems. There are many situations where decisions form part of the description of these systems.

Use Cases

The SysML Use Case can be used to describe a goal a user is trying to achieve by using the system. The Use Case can be described using a series of steps typically creating a backward and forward interaction between the user and the system. The steps the system performs often require decisions to be made and these can be modeled using a Decision Model. Consider a Use Case that describes an aspect of a Ship's navigation System. A step in a Use case could be 'The system decides on the optimal course and way points'. There would typically be a number of inputs into this decision which could be conveniently recorded in a Decision Model.



Activities and Actions

The SysML Activity diagram is a close (but older) cousin of the BPMN Business Process diagram and uses similar notation and semantics. Traditionally, decision logic has been intertwined with process flows using Decisions, Merges, Forks and Joins to describe the choices and conditions under which decisions are made. This has lead to complex and often unwieldy Process diagrams. Using the DMN, the decisions (including their logic) can be removed from the diagram and placed in a Decision Model. This has the effect of simplifying the diagrams, resulting in straight-through process flows and a model where decisions can be reasoned about, easily changed, and also generated to implementation code.

Unified Modeling Language

The Unified Modeling Language (UML) has become the de facto standard for modeling business- and software-centric systems. The types of system that are modeled using UML commonly have important decisions that form part of their specification and implementation. There are a number of places where decision modeling plays an important role.

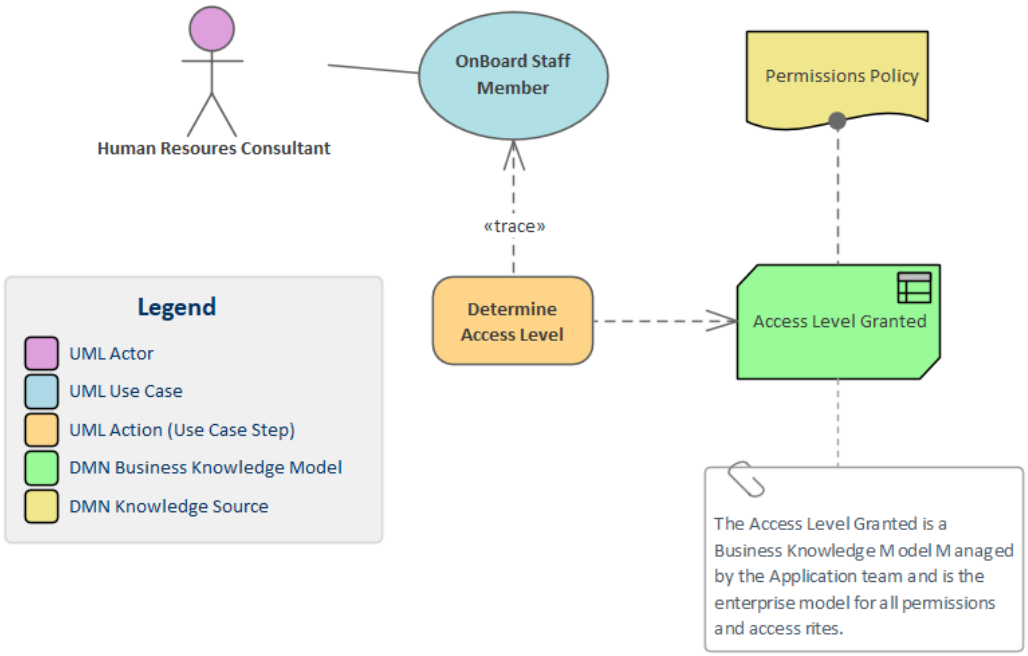
Activities and Actions

The UML Activity diagram is a close (but older) cousin of the BPMN Business Process diagram and uses similar notation and semantics. Traditionally, decision logic has been intertwined with process flows using Decisions, Merges, Forks and Joins to describe the choices and conditions under which decisions are made. This has lead to complex and often unwieldy Process diagrams. Using the DMN, the decisions (including their logic) can be removed from the diagram and placed in a Decision Model. This has the effect of simplifying the diagrams, resulting in straight-through process flows and a model where decisions can be reasoned about, easily changed, and also generated to implementation code.

Use Cases and their cousins User Stories

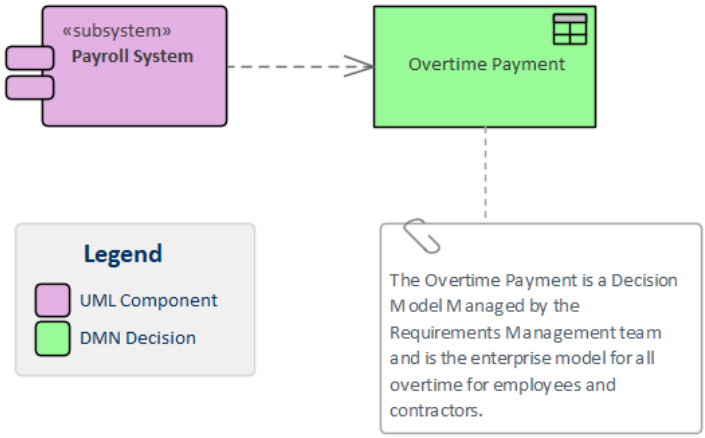
Although there are often robust debates about the differences between Use Cases and User Stories, they are both concerned with a goal that a User is trying to achieve. Many of these goals require decisions to be made at various points in the Use Case or User Story. In the example of Use

Cases, a Decision Model could be used to describe a system step in the Use Case, such as 'The system determines the level of access to give the user'.



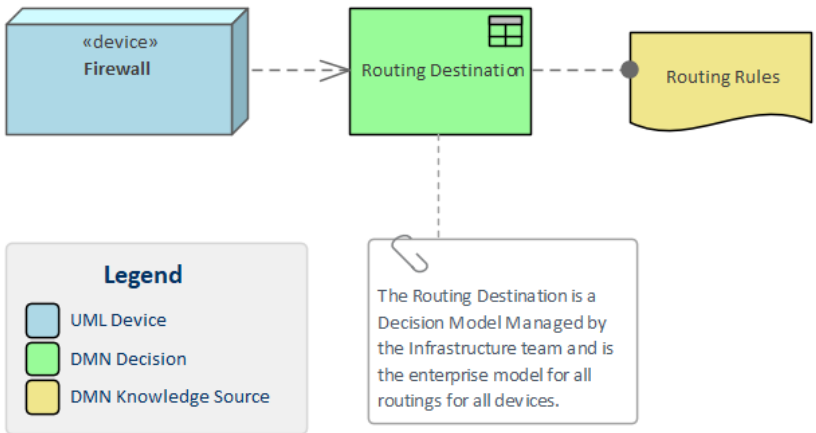
Components

Many systems are partitioned into a series of Components that are responsible for a discrete part of a system's function or services. In order for a Component to carry out its work, it is frequently required to make decisions. Consider a Payroll system that has to determine if over-time is applicable in a particular situation, or an Air Traffic Control system that has to decide whether to put an in-coming aircraft into a holding pattern, and for how long. (Most people at one time or another have been on the receiving end of that decision!)



Devices

Whether virtual or physical, many devices are required to make complex decisions. Consider a router that has to make complex decisions about where to route network traffic, or a traffic controller that has to schedule various traffic control mechanisms to optimize traffic flow, or a firewall that protects an organization's network.



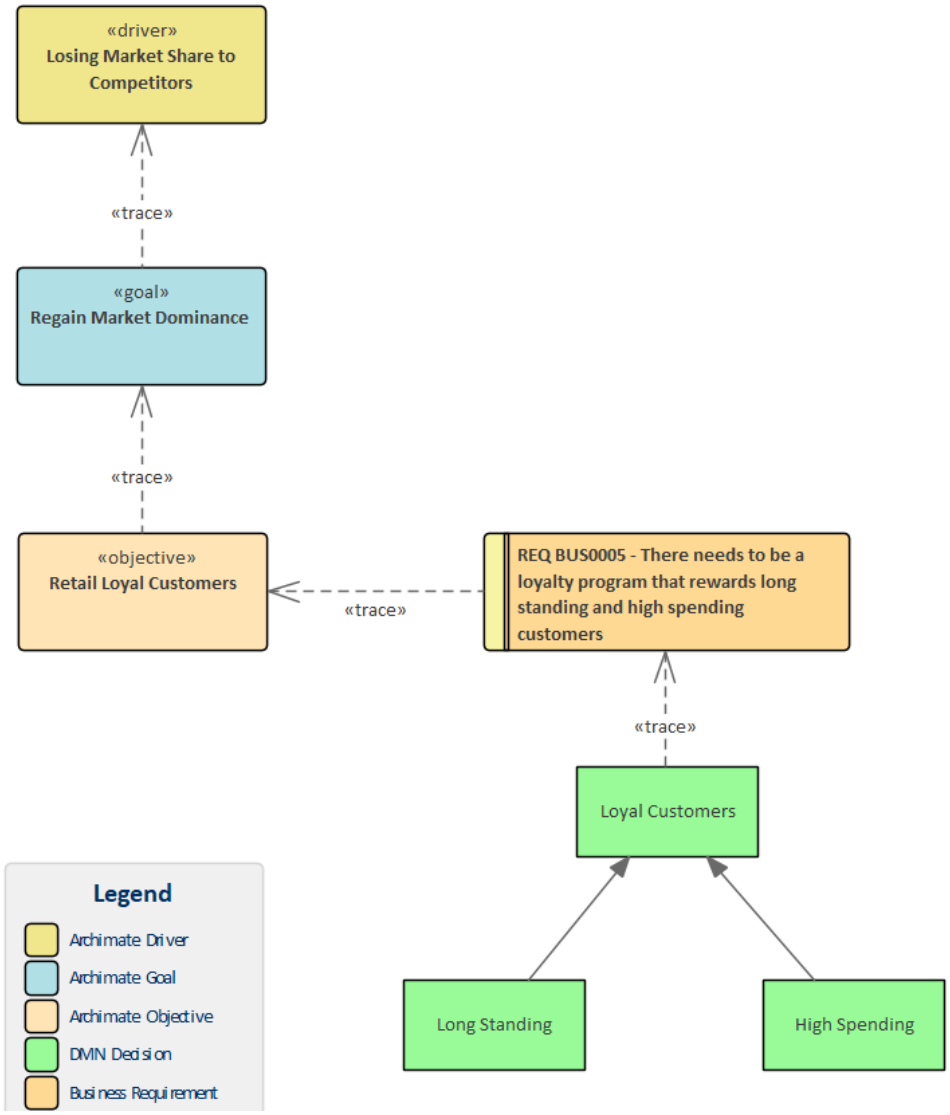
ArchiMate

ArchiMate is an enterprise architecture modeling language used to create, manage and visualize architectures at a number of different levels. There are a number of important

places where decisions can be defined and described, including at the level of strategy. Consider an architecture that defines an Application Service that selects a default set of products to offer to a customer. The decision about which products to bundle could be expressed in a Decision model.

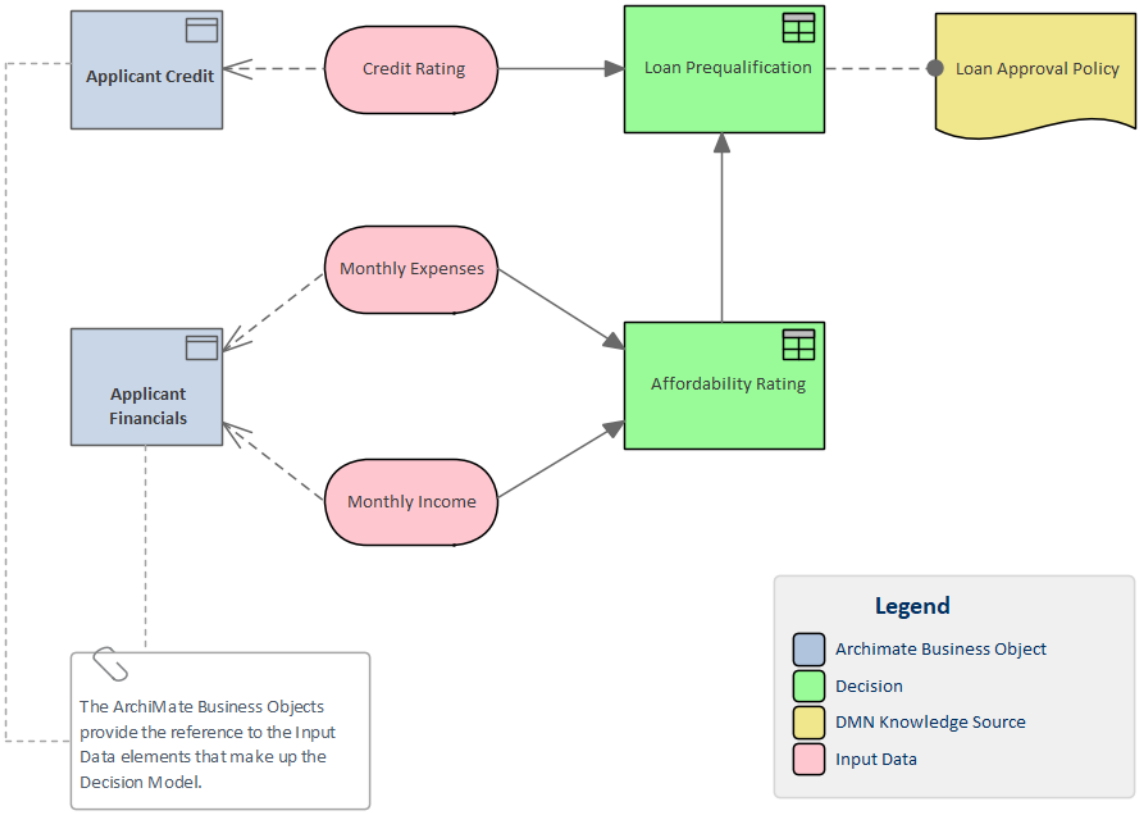
Drivers and Goals

Decisions can be related to drivers that create, motivate, and fuel the change in an organization. To fully articulate goals, decisions can be used to show potential differences in direction setting. There are often high level decisions that need to be made at this level of an organization.



Enterprise Information Models

Input Data required by Decision models can be related to entities in Information models at any level of detail, from high level conceptual models down to physical data model schemas. Connecting the Decision models with the Information models ensures that the data required by the decision is available when it comes to implementing the decisions.



Policies and Standard Operating Procedures

Decisions, Business Knowledge Models or Knowledge Sources can be related to elements that model polices, standard operating procedures or work flows. These are often the source of information that dictates or guides decisions.

Application Service

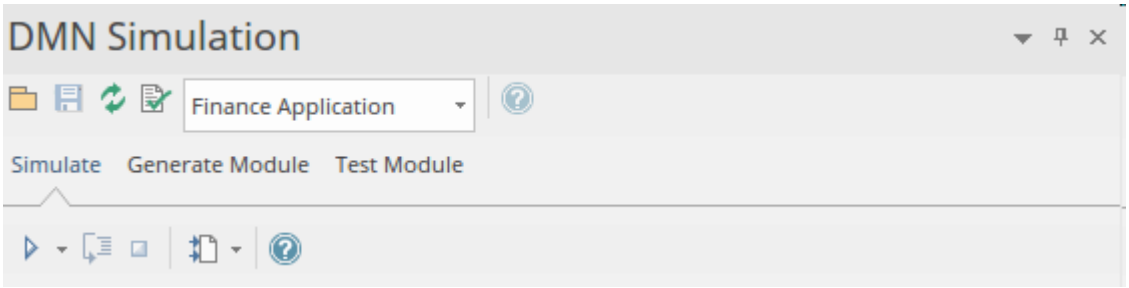
Decisions related to the provision of the service can be related to the Application Services to demonstrate how a service makes its decisions.

Simulating a Decision Model

Most business and technical stakeholders will be familiar with the high cost of correcting errors found in production systems. There are famous and well know examples and metrics that illustrate the fact that the cost of fixing something before it is put into production is usually an order of magnitude (or more) less than the cost of a remedy when a system has been delivered. The astronomers who launched the Hubble Space Telescope know this only too well, so too will a company that fails to identify hundreds of fraudulent loan applications. The cost of wrong decisions can be catastrophic and crippling to a business, technology, engineering or scientific endeavor. It is with this in mind that the designers of Enterprise Architect built a simulation capability into the modeling environment for Decision Models and, for that matter, a range of other technologies and languages. This, along with the modeling, collaboration features and code generation facilities, makes Enterprise Architect the pre-eminent platform for Decision Modeling and management.

The simulations can be run on the models as they are developed, before they are put into production, after structural model changes, or after rules or example data sets have been updated. The simulation is essentially a way of running trial model executions for the entire model or individual decisions within a complex model, allowing a team or an individual stakeholder to view the inputs, visualize the execution path, and analyze the intermediate

decisions and the outputs for a given model or fragment of a model.



The simulation runs seamlessly without the need for a modeler to perform any configuration, and two levels of toolbars provide a number of useful options. The top toolbar in the header of the Simulation window provides these options:

- Package Icon: Allows a Package to be selected
- Refresh Icon: Allows the module to be reloaded
- Validation Icon: Allows validation to be performed
- Drop-down list: Allows any decision in the model to be selected as the starting point for the simulation

With the 'Simulation' panel selected in the lower part of the Simulation window, a toolbar provides a:

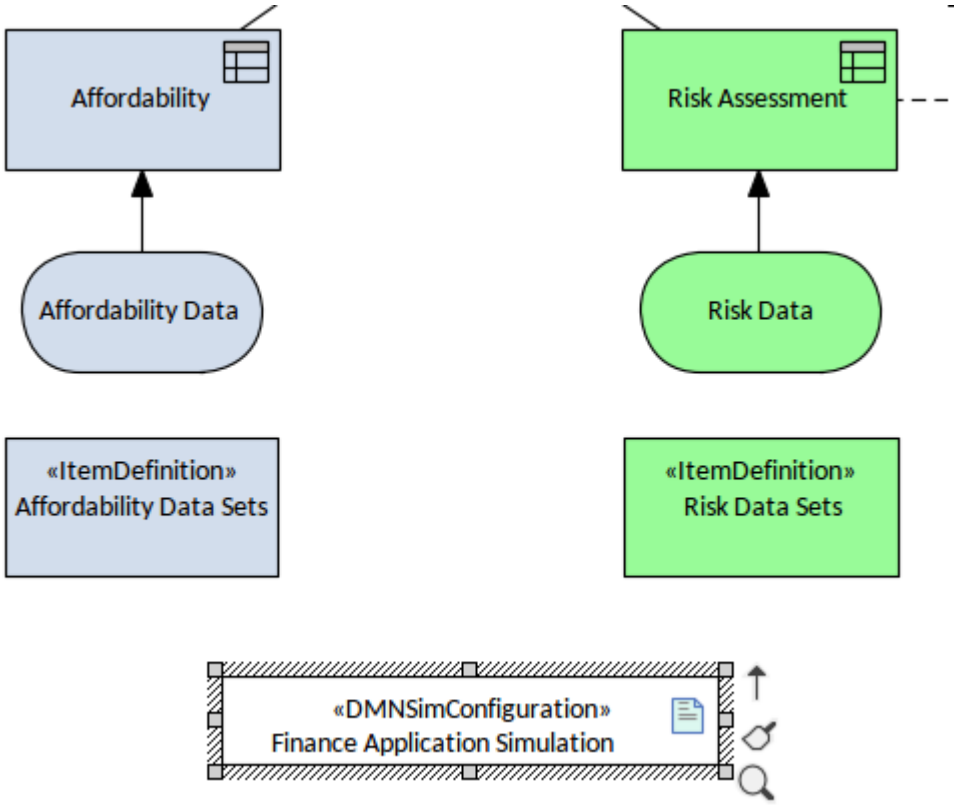
- Play Button: Run the simulation
- Step Through Button Step through each executable element in the model
- Stop Button: Stop the simulation during its execution
- Export Button: Export all or a selection of the Inputs to a BPMN Data Object

For more information see the DMN Simulation Toolbar topic.

Setting up a Simulation

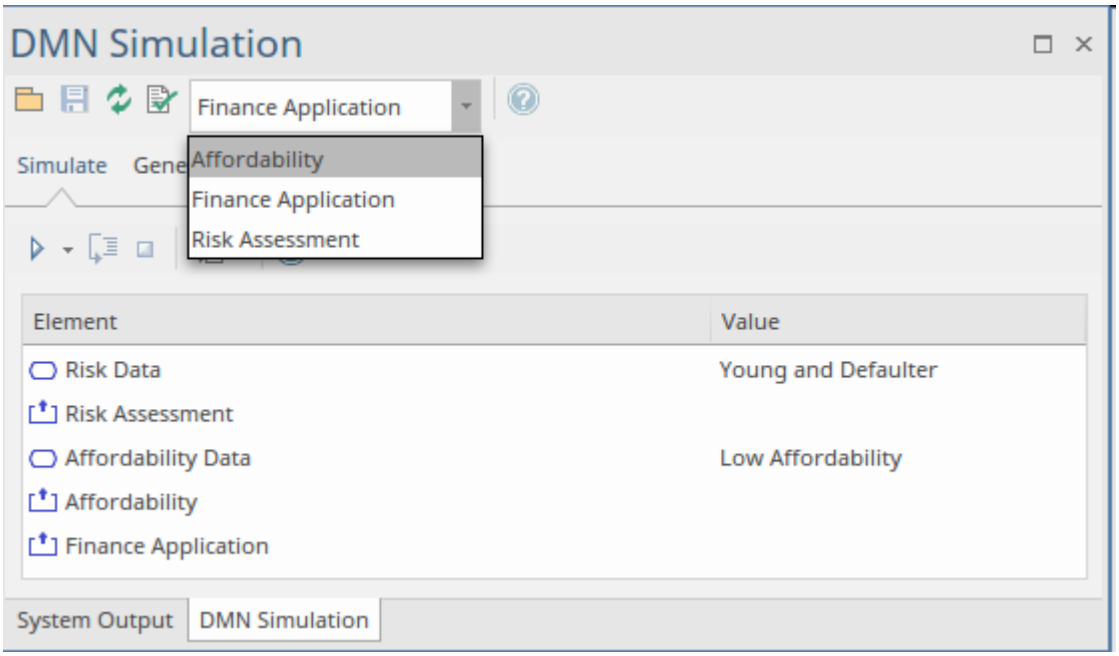
Setting up a simulation is straightforward; Enterprise Architect provides an Artifact that can be simply added to any Decision Requirements diagram to begin the process of configuring the Decision Model. The DMN Simulation Configuration element, available from the DMN Toolbox page, has the abbreviated name *DMNSimConfiguration*.

To start the process simply drag-and-drop a *DMNSimConfiguration* element onto a Decision Requirements diagram and then double click the element to launch the Simulation window.

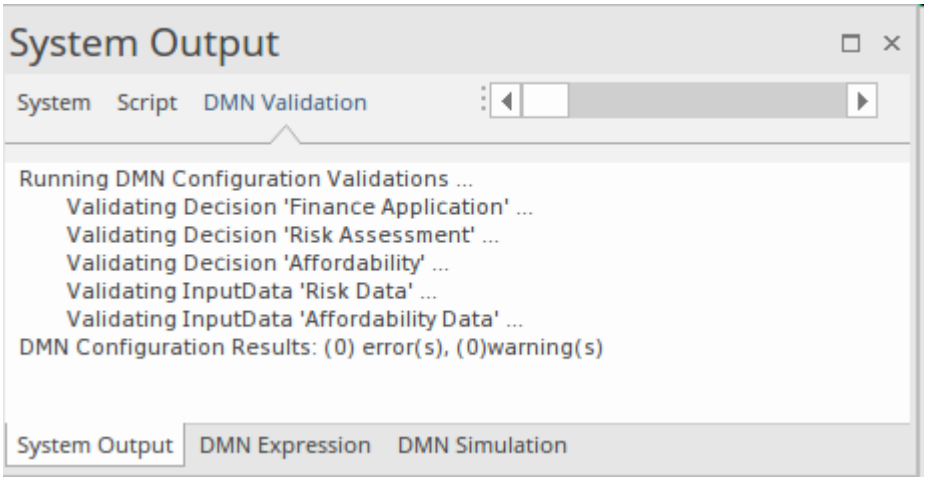


The Simulation window does most of the heavy lifting and there are only a small number of things that you need to do to configure the simulation. When the window opens it will

be preloaded with all possible Decisions or Business Knowledge Models that can act as starting points for the simulation; these are provided in a drop-down list from which you can select any one as the basis for the simulation. There are only two things that must be configured, and these are set to their defaults when you open the window - the selected decision defaults to the highest in the hierarchy, and the data sets are preset to their defaults. If you decide to work with the defaults, you can simply run the simulation without having to configure anything.

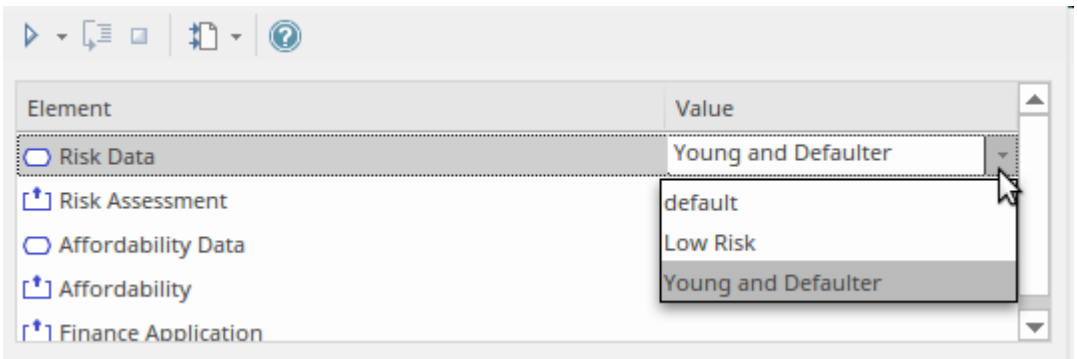


It would, however, be considered good practice at this point to run a validation check which would be run for all Decisions and Business Knowledge Models (BKMs) in the diagram. This option is available from the Simulation window toolbar and allows you to check that there are no syntactic or semantic (overlaps and gaps) in the expressions before running the simulation. If the validation results in errors it would be judicious to correct the problems before running the simulation.



The next step is the selection of the Input Data required for participant decisions and BKMs. This facility is one of the reasons Enterprise Architect is a market leader in this field, as it allows a modeler or a team to run the simulations with different input data that can be saved as a set and reused to visualize how the Decision Model will respond in different contexts.

A data set that has been predefined and given a meaningful name can be selected for each Input Data element. The list of data sets is available from a drop-down menu visible to the right of the Input Data item in the list. Selecting an item from the list tells the simulation engine that this is the data you want to use for the particular Input Data item and this will be displayed when you run a simulation.

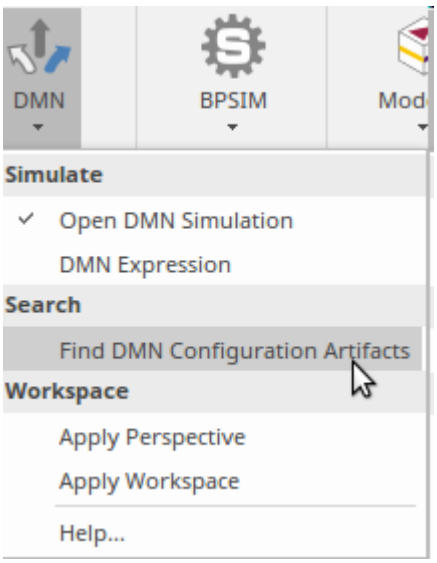


With these set you can run the simulation and view the

outputs. We will explore the features available for simulation in the next topic.

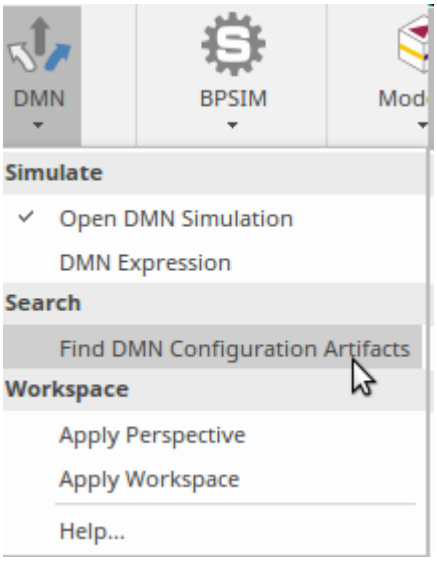
Finding Simulation Configurations

There might be occasions where you have forgotten the location of a Decision model that you have previously set up for simulation; in this situation Enterprise Architect provides a useful feature that allows you to locate the Simulation Artifact and therefore the diagram that contains it. The facility is available from this location:



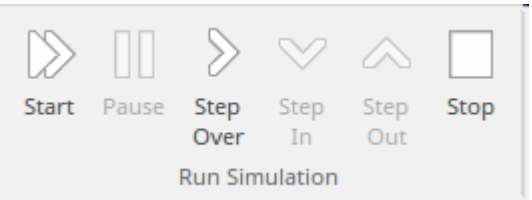
Simulate > Decision Analysis > DMN > Find DMN Configuration Artifacts.

This will return a list of Simulation Artifacts; using the context (right-click) menu you can locate the diagram that contains the element, and then by double clicking on the *DMNSimConfiguration* element in the diagram the Simulation will be launched.

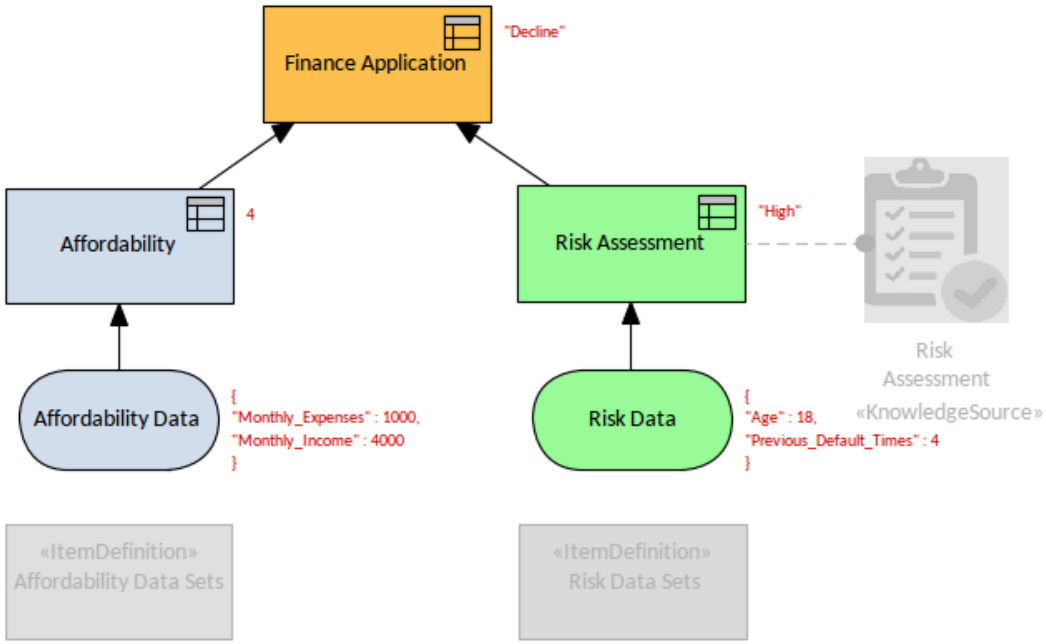

DMN
BPSIM
Mod
Simulate
Open DMN Simulation
DMN Expression
Search
Find DMN Configuration Artifacts
Workspace
Apply Perspective
Apply Workspace
Help...

Running a Simulation

The process for running a simulation could not be easier - you simply choose the Play button on the DMN Simulation window or on the 'Run Simulation' panel, available from the Simulation ribbon. It is common practice to have the Decision Requirements diagram open, but even if it is not Enterprise Architect will launch the diagram, as it will be used as a canvas for describing the decision steps as articulated in the next section.



Whichever method you choose, the simulation will run to completion and the results - including intermediate decision steps and input data - will be annotated on the diagram. These annotations will be welcomed by decision modelers and other stakeholders as they allow these stakeholders to visualize the mechanics of how the simulation engine arrived at the final outputs.



In this diagram you can see that the decisions have been executed from the root level up to the trunk of the hierarchy, and the final output of the highest level decision is *'Declined'* - meaning that the customer's application for finance has not been accepted. Even in this relatively simplistic model it is useful to be able to see the intermediate decision output values (such as Affordability=4, Risk Assessment = High), but in a complex model it becomes critical to be able to visualize the steps that contribute to the final decision output. There are a number of scenarios where this information is important:

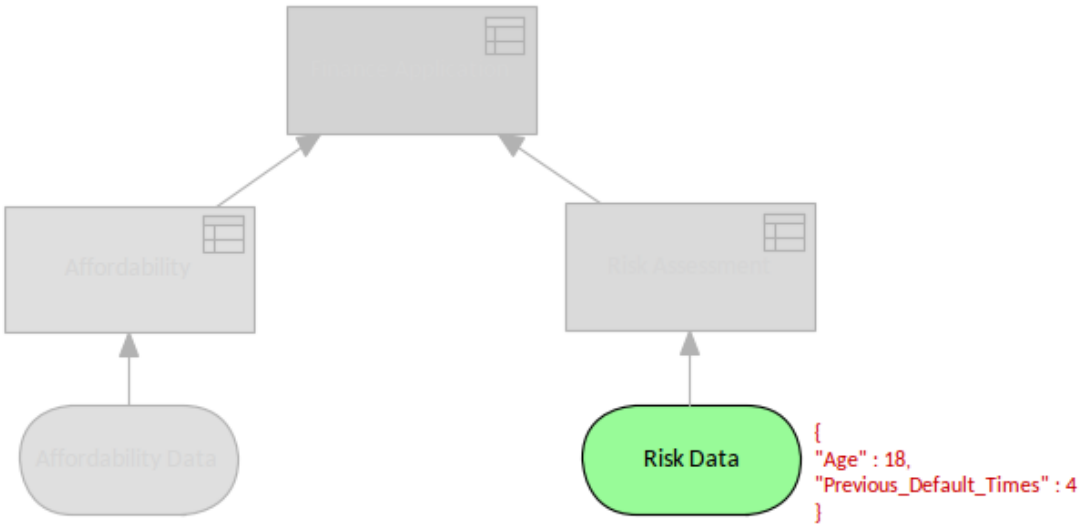
- Incremental Development of the Model: *including refactoring it after it has been deployed*
- Testing the Model: *to ensure that it is generating the correct results with given data sets*
- Decision Explanation: *including explaining to stakeholders such as customers how the decision was*

arrived at

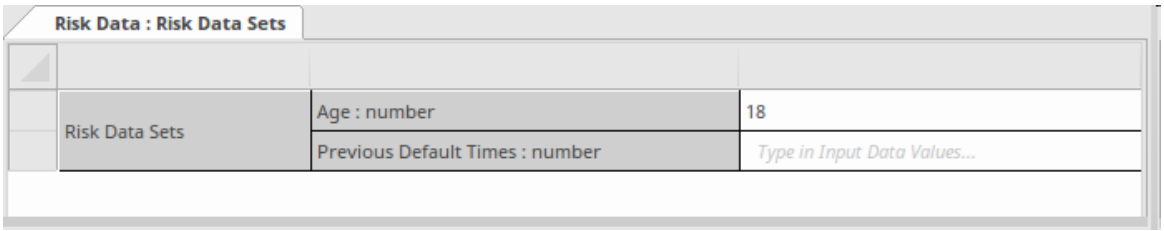
Stepping through a Simulation

An even more effective facility is the ability to step through a simulation, allowing you to effectively look over the shoulder of the engine as it is performing the simulation. Again, this will be welcomed by modelers as it has the added benefit of highlighting the rules in the Decision Tables as they are invoked, thus allowing the simulation audience to see exactly which rules were fired to arrive at the output at each step during the execution.

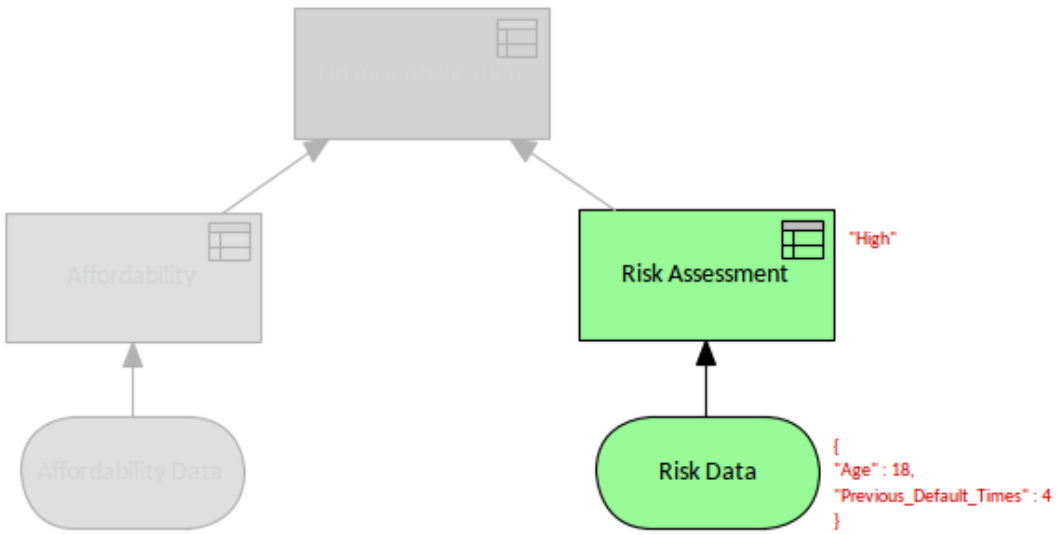
It is a remarkably useful feature and it is amazing how many business logic errors can be picked up at this time, allowing the rules to be refined and honed so they are deemed complete and correct before they are put into a production system. To step through the simulation you must run the simulation to completion by clicking on the Play button, as indicated earlier. Once the simulation has been run,you can then select the 'Step' icon and the engine will start the simulation from the first step and pause before the second step, allowing you to view the results step-by-step. This diagram shows the diagram annotation after the first step:



And this diagram shows the data set that had been configured for the input data and that is being used for this step in the simulation:



This information is also annotated on the diagram and can be seen after every step of the simulation as the information on the diagram accumulates, as each step is run.

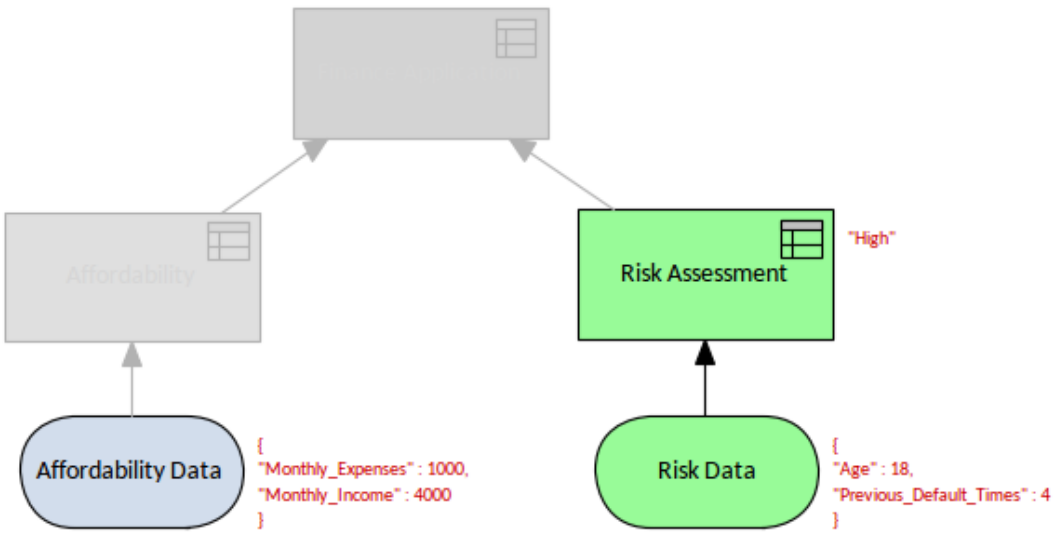


As you are stepping through the simulation, Enterprise Architect will display the selected rules in the expression editor and - in the case of a Decision Table - the rules that were fired will be highlighted in the table, allowing the simulation audience to see the rules clearly. The simulation will be paused at this point and will wait for you to click on the Step button again to resume the simulation.

Risk Assessment

U	Risk Data . Age	Risk Data . Previous Default Times	Risk Assessment
	18	4	*High*
1	<18	>3	*High*
2	<18	[2..3]	*Moderate*
3	<18	<2	*Low*
4	[18..55]	>3	*High*
5	[18..55]	[2..3]	*Low*
6	[18..55]	<2	*Moderate*
7	>55	-	*Moderate*

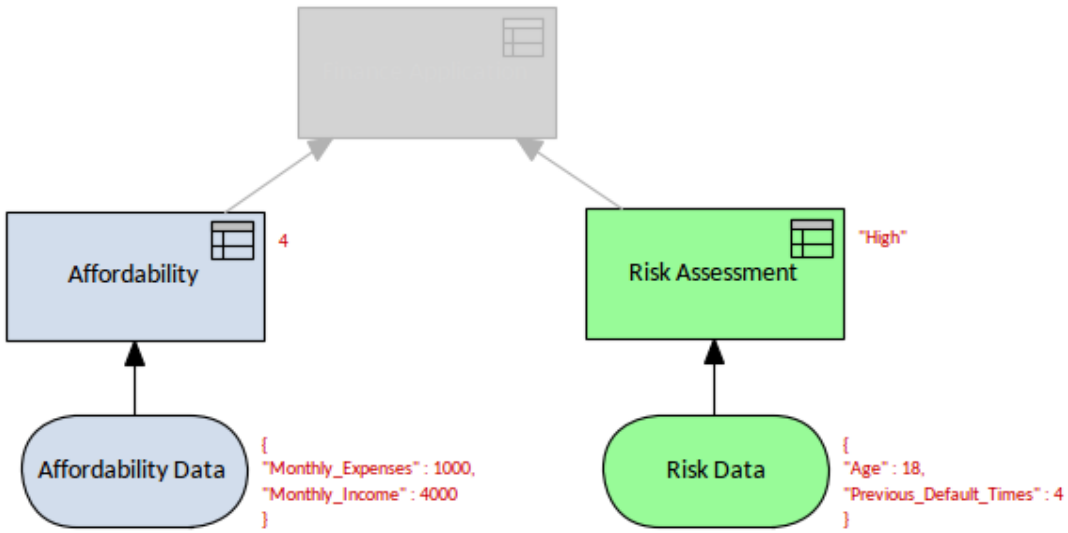
The simulation can be continued step-by-step, and the input data, the selected rules and the diagram annotations can be viewed for each step in the paused state before moving on, until the final result is output for the highest level decision in the decision hierarchy.



In this diagram the affordability data has been input by the simulation engine, and this will be used to calculate whether the applicant can afford to service the loan. We will see by the result in the next illustration what decision will be made.

Affordability

U	Affordability Data . Monthly Income	Affordability Data . Monthly Expen...	Affordability
	4000	1000	4
1	>10000	>3000	10
2	>10000	<=3000	9
3	[5000..10000]	<=2000	8
4	[5000..10000]	>2000	7
5	<5000	<=1000	4
6	<5000	>1000	3

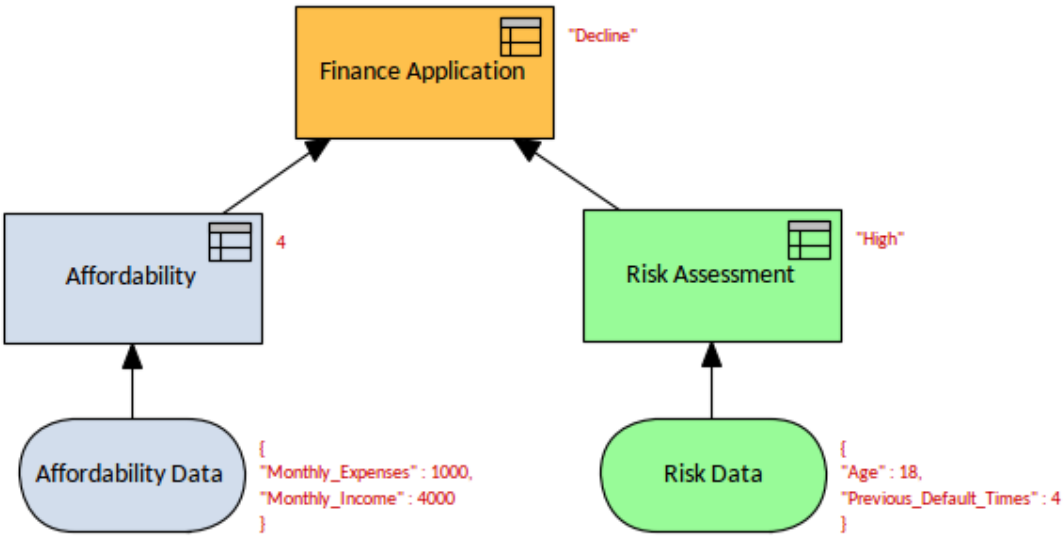


You can continue stepping through the simulation, and now we will show the result for the highest level decision and the rule that was selected in the Decision Table.

Finance Application

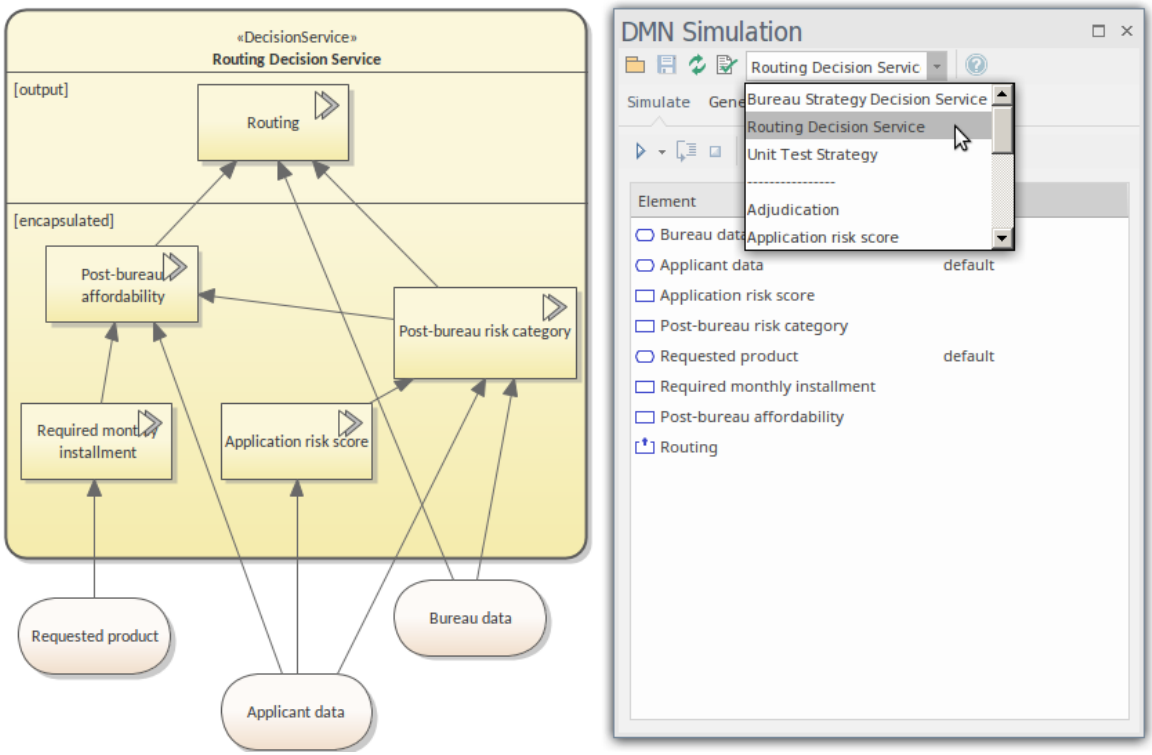
A	Risk Assessment	Affordability	Application Decision
	High	4	*Decline*
1	*Low*	>8	*Accepted*
2	*Moderate*	[5..8]	*Accepted*
3	*High*	<5	*Decline*
4	*Low*	[5..8]	*Accepted*
5	*Moderate*	>8	*Accepted*
6	*High*	[5..8]	*Review*

The next diagram shows the final result of the simulation, and unfortunately the Applicant's finance application would be declined using the data sets provided. This is indicted by the final output annotation to the right of the Finance Application decision, which is the highest level decision in the hierarchy.



Simulating a Decision Service

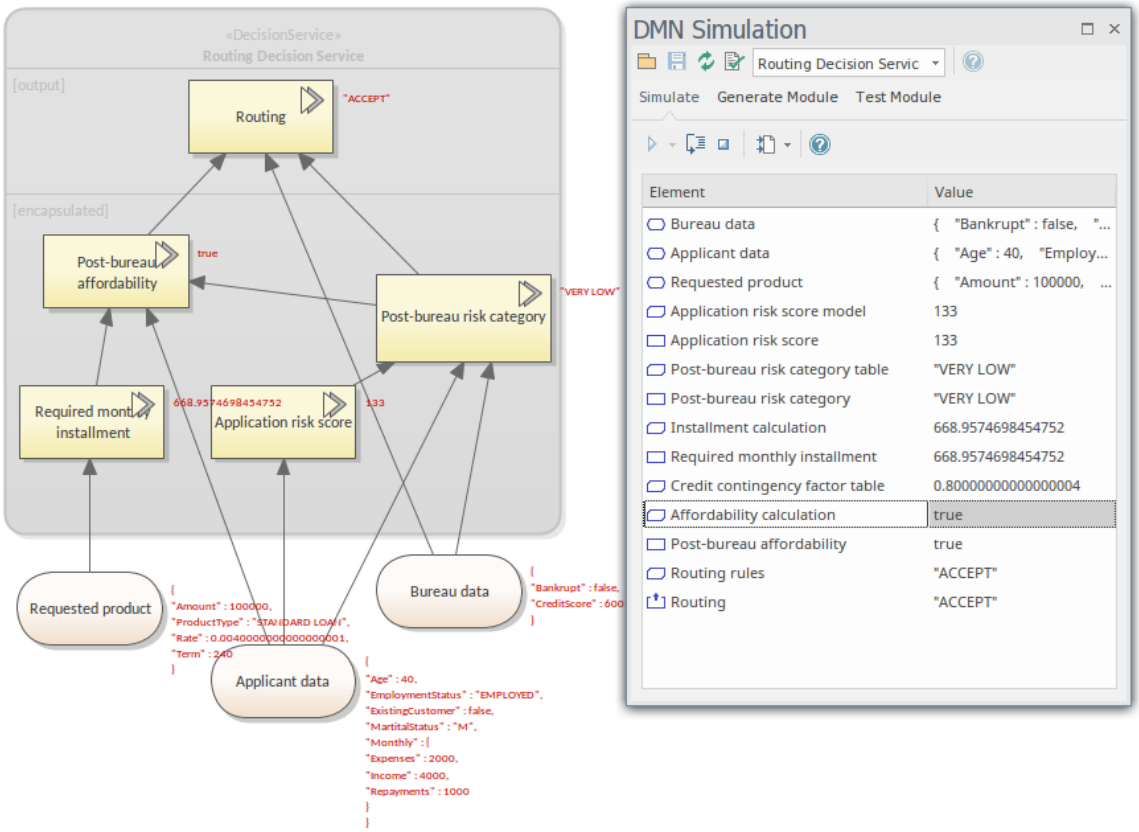
Decision Services can be simulated in the same way that other decisions are simulated. The starting point is as always to drag-and-drop a DMN Simulation Configuration Artifact onto the diagram, and then select the Decision to act as the starting point and any Input Data sets required.



By default, all Decision Service elements and every discrete Decision are listed for selection in the drop-down field in the dialog toolbar.

The input data and decisions are in the correct execution order. For example, 'Application risk score' will be executed before 'Post-bureau risk category', 'Post bureau affordability' and 'Routing'. This follows the principle of the decision hierarchy, which starts at the roots and moves to the trunk (the highest level Decision that has been selected). After

providing the input data by choosing a data set in the combo box, click on the Save icon and on the button on the toolbar.



This diagram shows the result of the simulation, which allows the simulation audience to visualize how the final decision was arrived at by annotating the diagram with the intermediate decisions and input data. It is also possible, once the simulation has been run, to step through the simulation and see how each decision was reached, including the visualization of the rules and expressions in the Decision Expression Editor. For example, if an expression has been defined using a Decision Table, the rule or rules that are hit will be highlighted allowing the audience to understand the logic as it is applied - and potentially correct errors or oversights.

Code Generation from a Decision Model

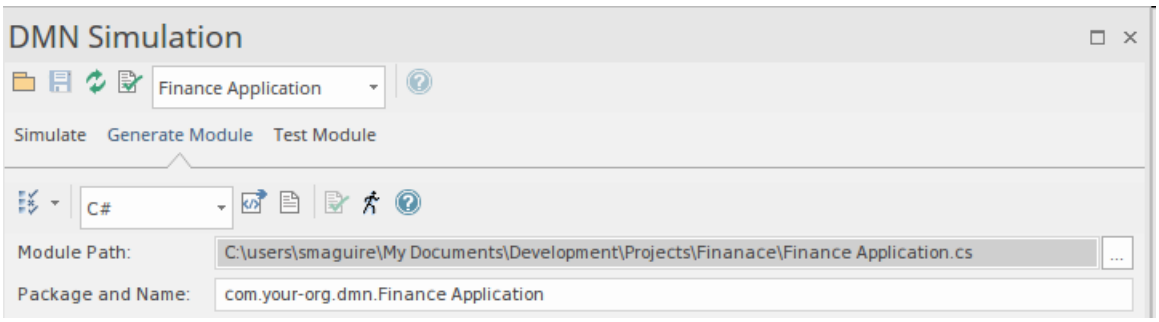
Many of the errors that creep into production systems are the result of the misinterpretation of business intention by the technical staff charged with the implementation of a system. These errors are not intentional and even with the best processes in place they can inadvertently find their way into critical systems. The possibility of errors is minimized and almost completely eradicated by Enterprise Architect's Code Generation facility.

This feature allows the straight-through generation of implementation code from the models that have been created, validated and simulated and are asserted to be correct. The code is generated automatically by Enterprise Architect with no need for human intervention, and the target can be in any one of these programming languages:

- JavaScript
- Java
- C++
- C#

The choice of language will depend on the target system. The generated code can be used in a number of different contexts including:

- BPSim Execution Engine - an Enterprise Architect facility to simulate Business Process models
- Executable StateMachine - an Enterprise Architect facility

that executes StateMachine models

- Your organization's proprietary implementation environment deployed as a platform or service