ENTERPRISE ARCHITECT

User Guide Series

Systems Engineering

Author: Sparx Systems
Date: 2026-05-04
Version: 17.1

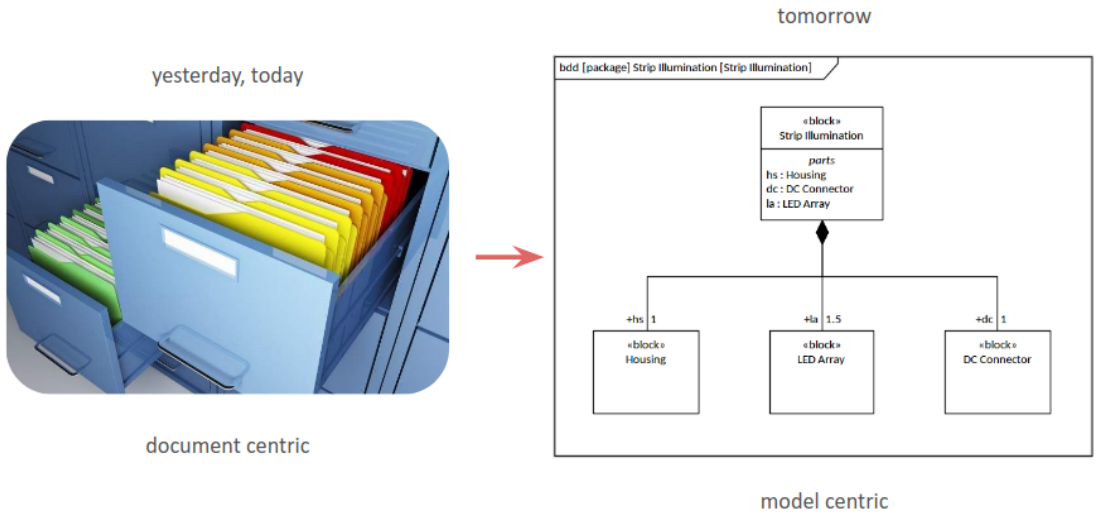
CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

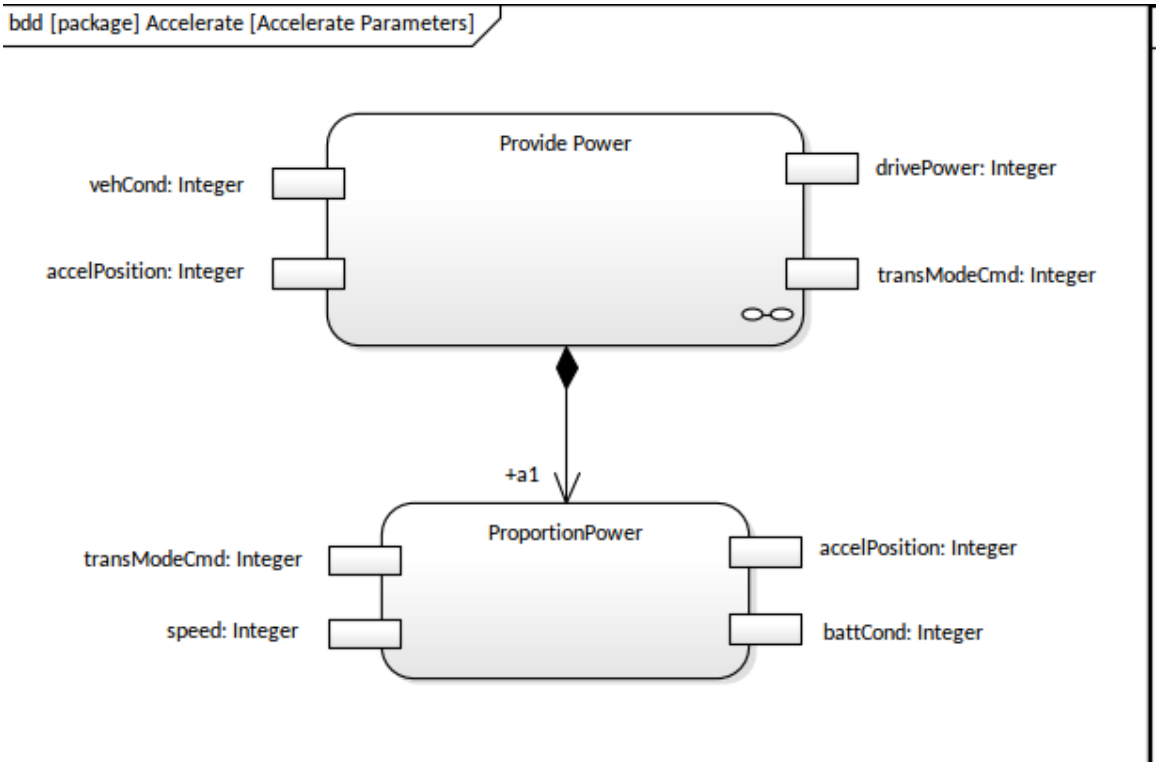
Systems Engineering

Systems Engineering is, very broadly, the work of researching, designing and managing complex physical or electronic systems over their lifecycles. It focuses on the whole system and typically involves a number of sub-disciplines such as requirements, reliability, logistics, design, testing and maintenance; it considers not only the system itself but also processes, optimization and risk management, and requires sophisticated project management techniques.

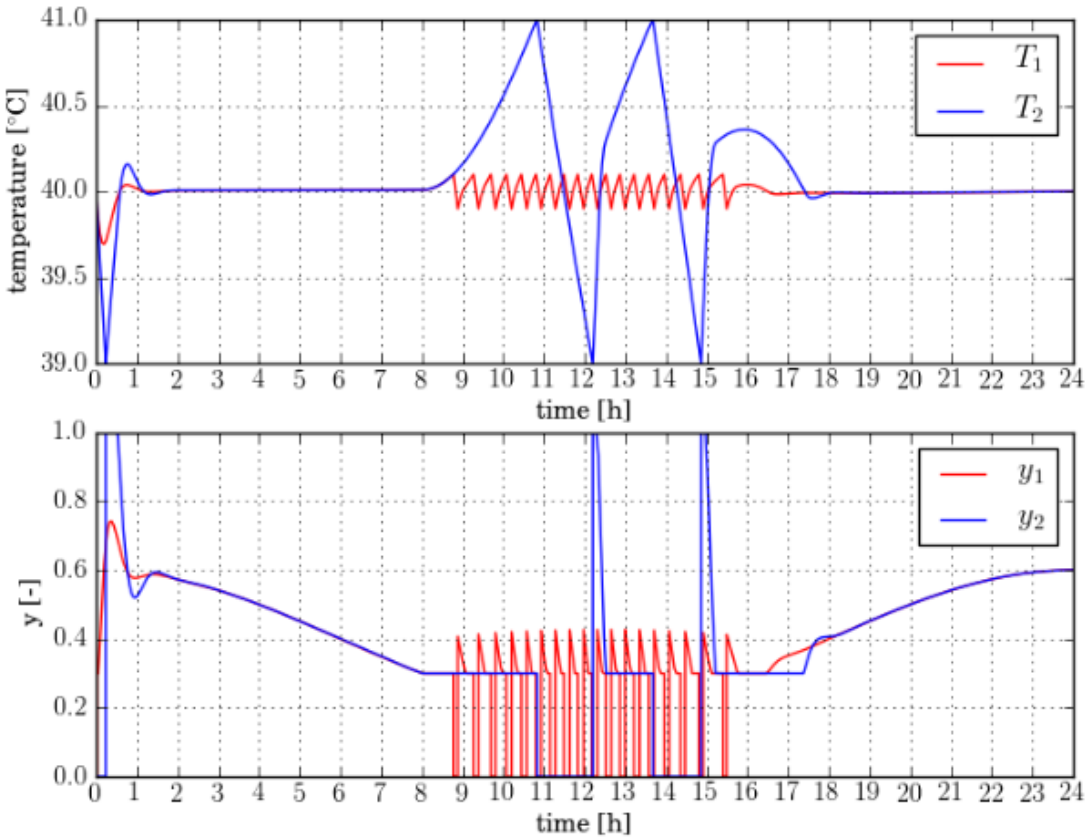
In earlier decades a large but localized team might consider a very specific set of objects within a very specific and controlled environment, to be delivered to a small user base and maintained by perhaps an, again, localized team of experts who each might have responsibility for only a part of the system. Even for such a controlled and structured scenario, a huge volume of documentation was required to define the system requirements, the components, the engineering process, the standards applied and complied with, and the tests to be run on the system. Keeping this documentation cross-referenced, up to date and integrated was a major task.



Advancing into computing and basing Systems Engineering work on graphical models (Model Based Systems Engineering) provided huge benefits, allowing engineers to store and retrieve data from repositories, associate data with documentation also held in the repositories, and develop both master structures and variants from templates, all of which reduced the need to recreate and repeat work. The model initially represented the organization of the developing system, but grew to reflect the development process and the factors that supported and directed that process. As computing capabilities grew, and more specialized and sophisticated applications were made available, it became possible to represent the components of a system with increasingly varied and detailed model elements, and with increasingly varied and detailed relationships between them.



Engineers could 'load' the model components and relationships with an array of properties, characteristics and parameters, which could be varied to reflect different scenarios. The standards that the system must apply or meet could be automatically enforced on the components as constraints, conditions and rules. More and more of the development process - such as testing - could be represented by element or model features, and more and more aspects of the process could be performed on the model by the application - such as automatic generation of code to make the system operational, and simulation of the system in action under various conditions.



Currently, the Systems Engineer is likely to be a member of an interdisciplinary team that has to consider a wide range of factors in defining or applying an architecture and then designing and modeling a system - a much broader, diverse and inexpert user base, a much broader maintenance base, how the system interacts with many other systems, how the system operates in many different and sometimes extreme environments, the impact the system has on the global environment - both within its operating framework and within its pre-use production and final disposal - the socio-economic environment controlling its acceptability and popularity, and how the system compares with its increasing range of competitors. To see how the work of the Systems Engineer has become vastly more complex one has only to think of a single development, such as the quantum leap from the relatively recent fixed-site landline telephone handset for making voice calls, to the modern mobile smartphone used as a camera, computer, cinema, music center, navigator, and audio, visual and text communicator.

Today, large projects and industries are being developed around systems and products for which the use cases are increasingly complex. Controlling this complexity grows further and further beyond the capacity of the engineer, increasing the level of risk to the product, the end user and the manufacturer. Examples of systems with dramatically increased risk include the manufacture of passenger air bags to be fitted to many different brands and types of car manufactured in different parts of the globe; or the requirements for the development of space probes intended to travel to the planets of the solar system and beyond.

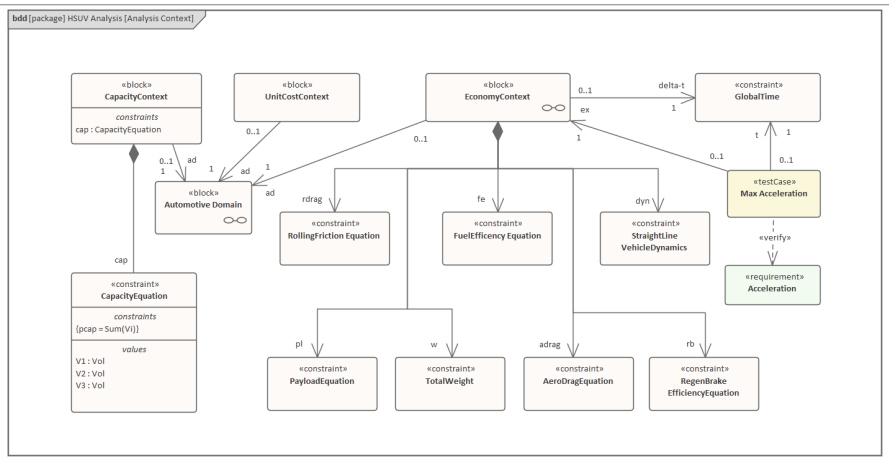
It is the advances in Systems Engineering tools and methodologies that have increased this complexity, whilst simultaneously providing the capability to manage and mitigate the associated risk, and reducing the difficulty and effort involved in managing and maintaining highly complex models.

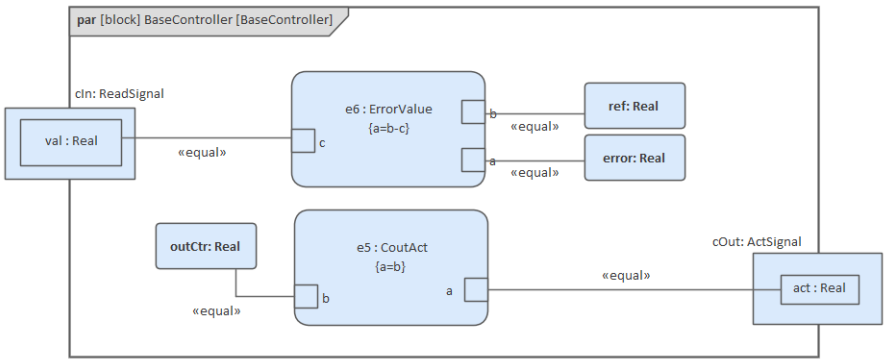
For additional information, see the *Representing Systems with Models* section of the 'SEBoK - Guide to the Systems Engineering Body of Knowledge' website.

Model-Based Systems Engineering in Enterprise Architect

Enterprise Architect provides a Model Based Systems Engineering platform that integrates many high-end features for Systems Engineers and model-based development, with these built-in features.

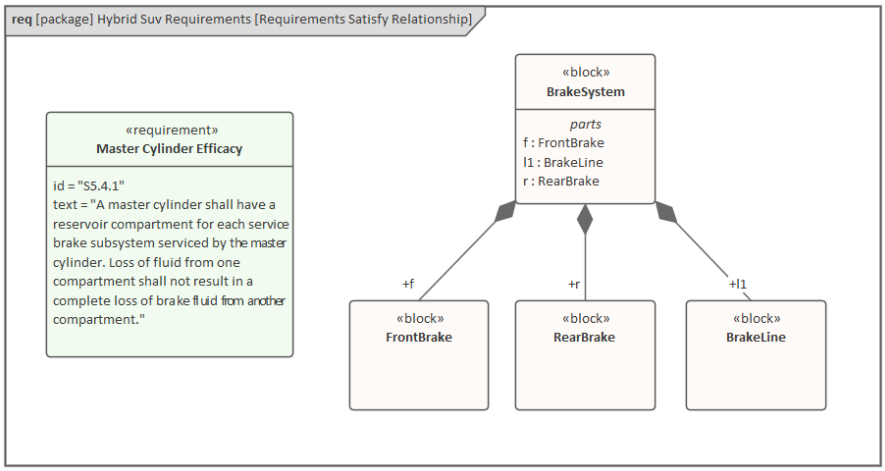
Feature	Description
SysML	Enterprise Architect is integrated with the Systems Modeling Language (SysML) versions 1.1, 1.2, 1.3, 1.4 and 1.5. For details, see the *Systems Modeling Language (SysML)* Help topic. Enterprise Architect provides a number of engineering model templates from which models of engineering structures and concepts can be developed. This is an image of a SysML 1.5 Block Definition diagram. It is part of the HSUV Model that can be found in the 'Systems Engineering' section of Enterprise Architect's Example Model.

	
Conformance to Standards	As well as applying the standards defined by the OMG for UML and SysML, the Enterprise Architect Model Based Systems Engineering platform also complies with these international standards: • International Council of Systems Engineering (INCOSE) 2012 • Ontology Definition Metamodel (ODM) (OMG document *ptc/2013-12-03*, pub. February 2014) • Systems Modeling Language (SysML) (OMG document *formal/2017-05-01*) • Unified Profile for United States Department of Defense Architecture Framework (DoDAF) and United Kingdom Ministry of Defense Architecture Framework (MODAF) (UPDM) (OMG document *formal/2013-01-01*)
Executable Code Generation	You can quickly generate executable software code from your model elements, using Executable StateMachines. The code generated for an Executable StateMachine is based on its language property. This might be Java, C, C++, C# or JavaScript. Whichever language it is, Enterprise Architect generates the appropriate code, which is immediately ready to build and run. There are no manual interventions necessary before you run it. For more information, see the *Code Generation for Executable StateMachines* Help topic.
Model to Code Transformations for HDLs	You can not only generate executable software code, but you can generate Hardware Description Languages and Ada from your model elements, for the chips and circuits in system hardware components. For more information, see the *StateMachine Modeling for HDLs* Help topic.
Parametric Model Simulation	Enterprise Architect provides facilities to create Parametric diagrams using the Parametric Diagram Modeling Assistant, and to perform Parametric Model Simulation through OpenModelica. Being able to simulate a system through the model is a huge advantage where live-testing would be dangerous (defense systems) or prohibitively expensive (space probes). This image shows an Internal Block diagram used in a Parametric Model Simulation. The diagram is part of the 'Two Tanks' example that can be found in the 'Systems Engineering > Modelica Examples' section of Enterprise Architect's Example Model.



For further information, see the *Parametric Diagrams*, *Parametric Diagram Modeling Assistant* and *Parametric Simulation Using OpenModelica* Help topics.

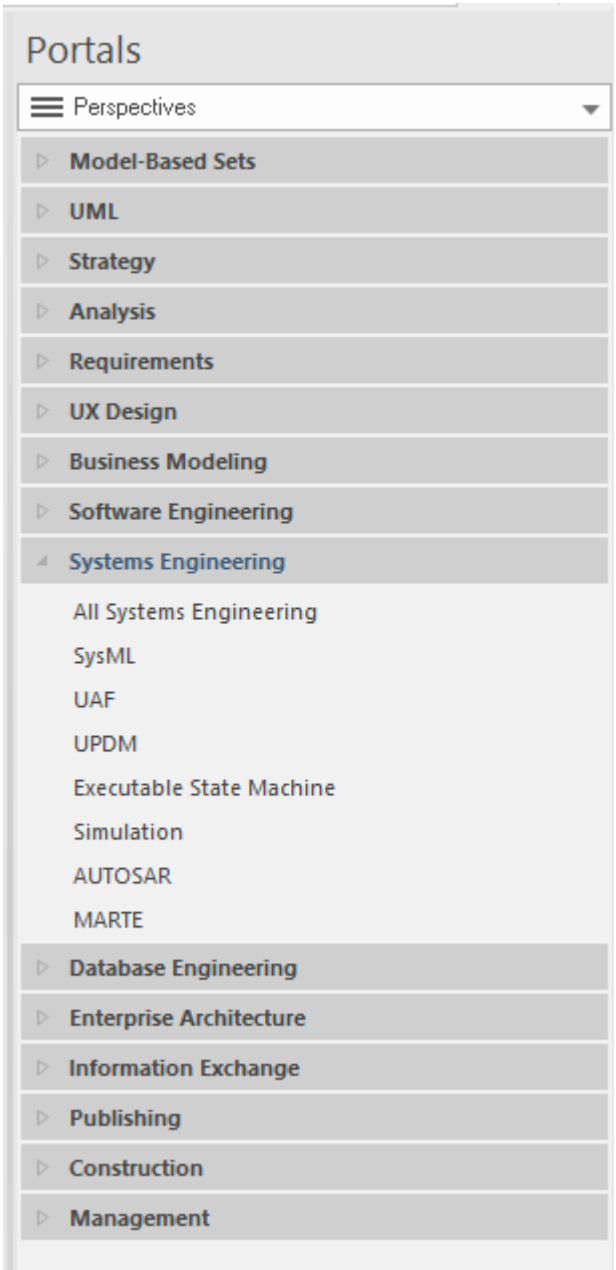
System-of-Systems Modeling	In addition to developing system models, you can also design 'system-of-system' models, or system architectures, using the Unified Profile for DoDAF and MODAF (UPDM) or the Unified Architecture Framework (UAF); these are both accessible through the Systems Engineering Perspective with SysML.
Requirements Management	Enterprise Architect has an extensive suite of Requirements Management tools that can be applied to System Engineering, dove-tailed to the SysML Requirements modeling facility. See the *SysML Requirements Model* and *Requirement Models* Help topics. This image shows an example of a SysML Requirements diagram.
Project Management	Enterprise Architect has extensive Project Management and team support facilities to help you organize, support and manage both the Systems Engineering model content and the staff working on the project. Amongst other things, you can apply user security, organize and monitor resources, schedule tasks, apply Version Control and enable a range of discussions from simple messaging through informal topic discussion threads to formal reviews. For more information, see the *Project Management* and *The Modeling Team* Help sections.



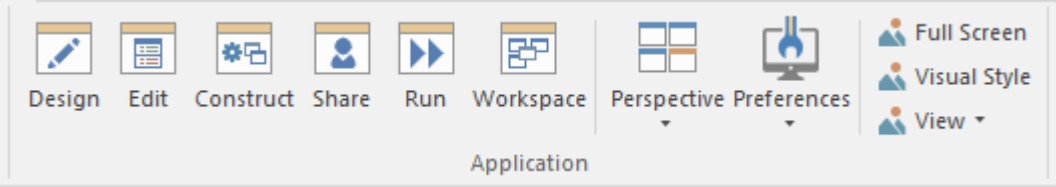
Getting Started

Getting started with a new tool can be quite daunting even for experienced engineers, 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 of its coverage might appear overwhelming to a person new to the program, but fortunately a solution to this has been built into the design. One of the main tool features is Perspectives.

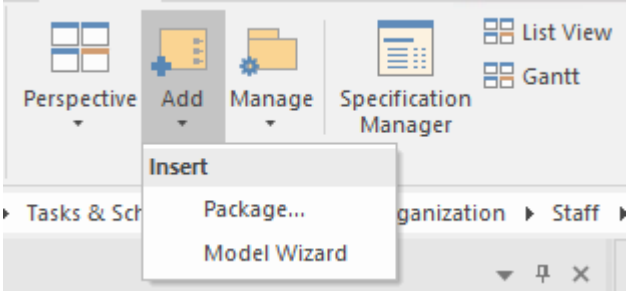
You can use Perspectives to limit the functionality to a specific domain or language, such as System Engineering, making it easy for a System Engineer or Manager to get started. A user still can utilize other functionality that might be useful, such as Strategic Modeling, Mind Mapping, Code Engineering 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.



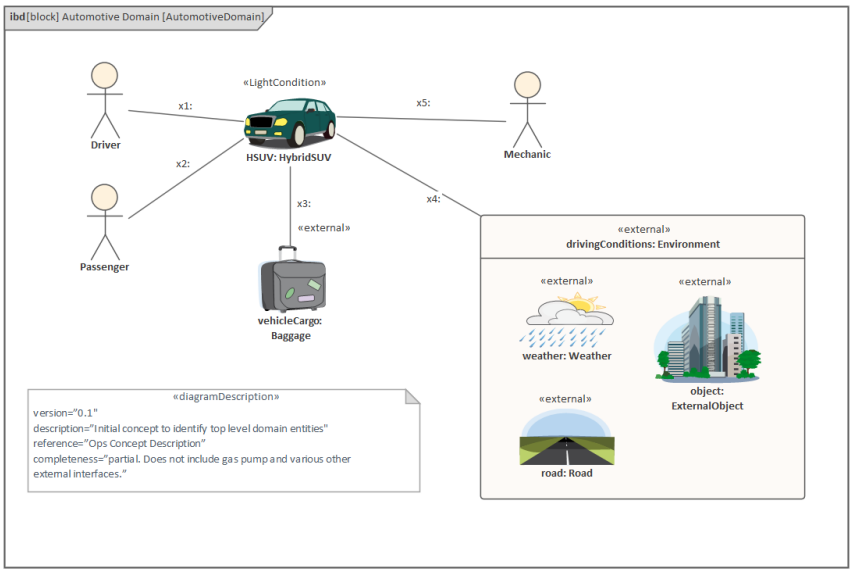
A user also has tremendous flexibility to tailor their environment and the user interface by setting preferences and selecting workspaces and visual styles.



Setting up a new project is straightforward with the use of the Model Builder patterns (with accompanying documentation) that can be utilized to automatically create an MBSE project structure to get you started. You can use the Model Builder to create any number of SysML diagrams as you develop the model and the problem and solution spaces are fleshed-out.

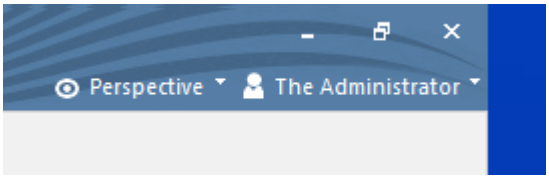


These and other facilities make it easy for a newcomer to get started, allowing them to become productive members of a team and start contributing to models quickly and without any delay. A novice engineer will be surprised how productive they can be when compared to working in text-based or other more rudimentary modeling tools. There will be challenges along the way as you push yourself and the tool to new limits but a detailed Help system, a large community of users, comprehensive forums, a community site and first-class support services will make the journey easy and informative. You will be able to create expressive diagrams like this one from the automotive domain, and communicate with engineering colleagues, managers, consultants and customers.

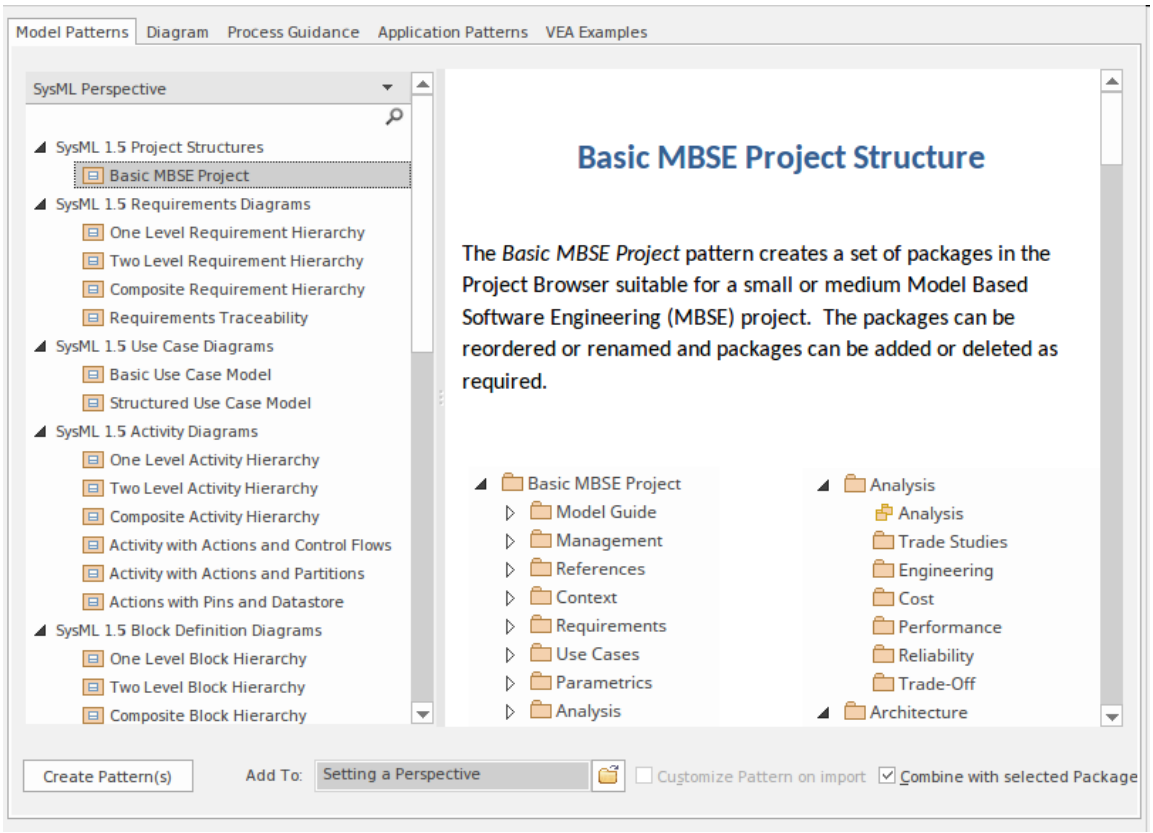


Setting a Perspective

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 tools features and facilities. The Systems Engineering group of Perspectives provides a natural starting point for Systems Engineers, but at any point if you decide to use other facilities in the tool you can simply change Perspectives and the tool will change to provide a focus on the selected area.

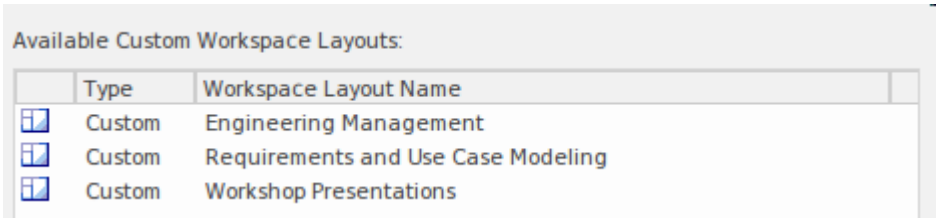


Selecting one of the Systems Engineering Perspectives will change the tools to focus on the selected aspect of Systems Engineering. For example, choosing the SysML Perspective will display a series of model patterns giving a user a jump start by being able to load a pattern for a standard model fragment or diagram. The 'Diagram Builder' dialog will also just display SysML diagram types.



Selecting a Workspace

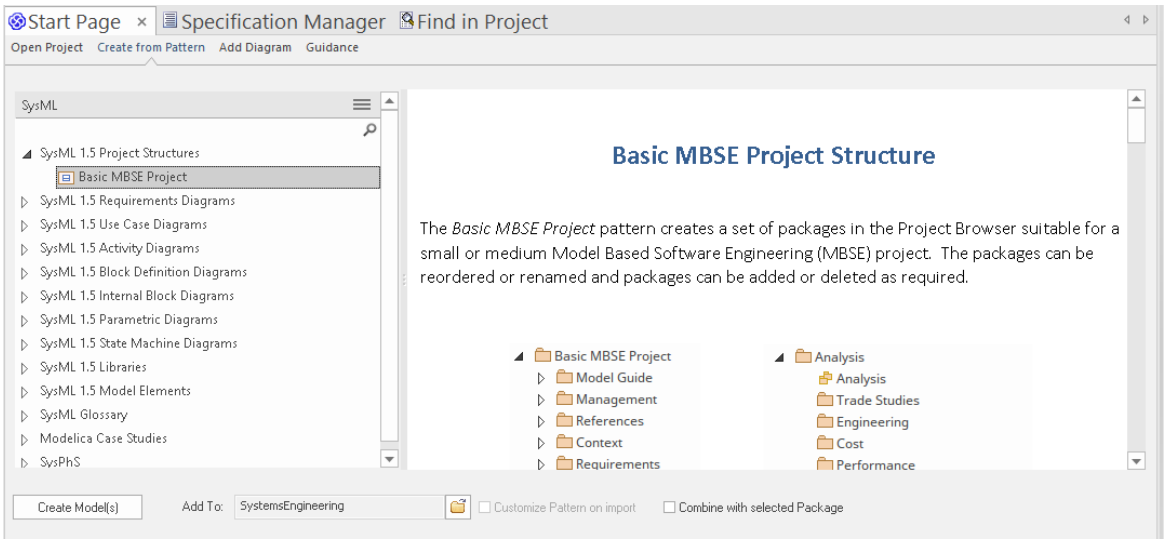
Enterprise Architect has a helpful way of quickly changing the User Interface layout to facilitate particular engineering or management tasks or ways of working. This is achieved by simply selecting a workspace that will change the visible windows and tools to provide the most efficient working method to suit the task. For example, there is a workspace defined for Systems Engineering Simulations, one for Use Case Modeling, and another for Testing. You can also specify any number of your own workspace layouts that you find helpful by opening windows and tools and positioning them in an arrangement that facilitates working on a particular task or set of tasks and saving them. In this example, a modeler has defined three custom workspace layouts.



Setting Up a Model Structure

Enterprise Architect has been designed as a productivity tool from the ground up. One of the first tasks in a modeling project is setting up a model structure which can sometimes be daunting for the beginner and tedious for the experienced user. Enterprise Architect makes this task simple by using the Model Builder.

You can create the structure for a new initiative (project) using the Model Builder, which will produce an entire project structure that can be tailored on import, providing all the Packages ready to start the project.



The repository structure is a subject that is explored in a later topic because it is critical to the success of a model-based engineering approach to Systems Engineering. We will learn later that Packages are essential units in the organization and maintenance of a model repository. There is an entire topic dedicated to using Packages to structure the repository. For more information, see the Model Builder Help topic.

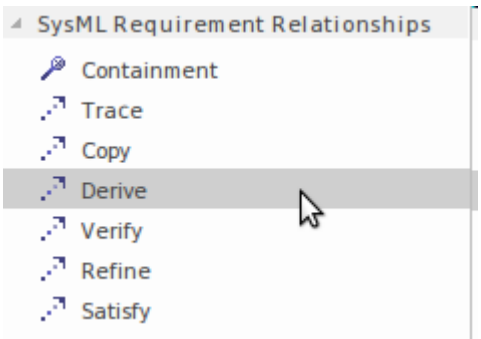
Example Models

Transitioning from a document centric approach to a model based systems engineering method can present some hurdles for the newcomer. Fortunately Enterprise Architect has rich and supportive help available and a series of in-tool facilities such as patterns that will get you started with the tool and project models. In this topic we present some straight-forward examples of requirement, structural and behavioral models and the diagrams that you or your colleagues would create in a typical project.

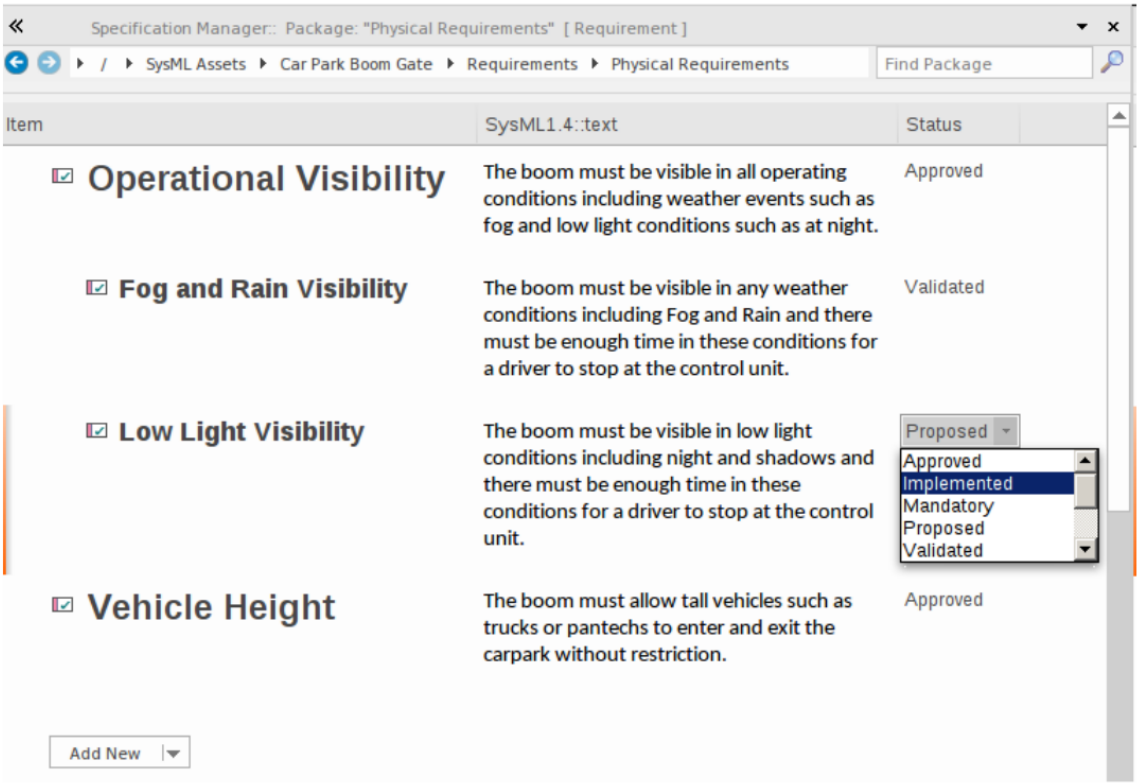
Requirements Models

Requirements models are fundamental to any Systems Engineering project whether be a greenfield project or a change to an existing system. Requirements are typically derived from a number of sources including meetings and workshops with stakeholders, documents or formal and formal requests from sponsors and other project stakeholders. There are a three main types of requirements including high level strategic or business requirements, user requirements and system requirements often referred to as quality attributes of the system.

Requirements models will typically evolve over the lifetime of the project and adaptive and iterative methods encourage requirement changes as stakeholders view the partially completed product at critical project milestones. These changes can be represented in the tool in a variety of ways including using the change management functionality available from the construct ribbon. Alternatively these changes can be represented as derivation relationships and visualized in a requirements diagram.



Enterprise Architect provides a rich suite of tools for requirement elicitation, development and management and enforces good requirement engineering practices. One of the key tools for working with requirements is the Specification Manager which allows requirements engineers more familiar with tools like word processors or spreadsheets to work in these familiar paradigms inside Enterprise Architect.

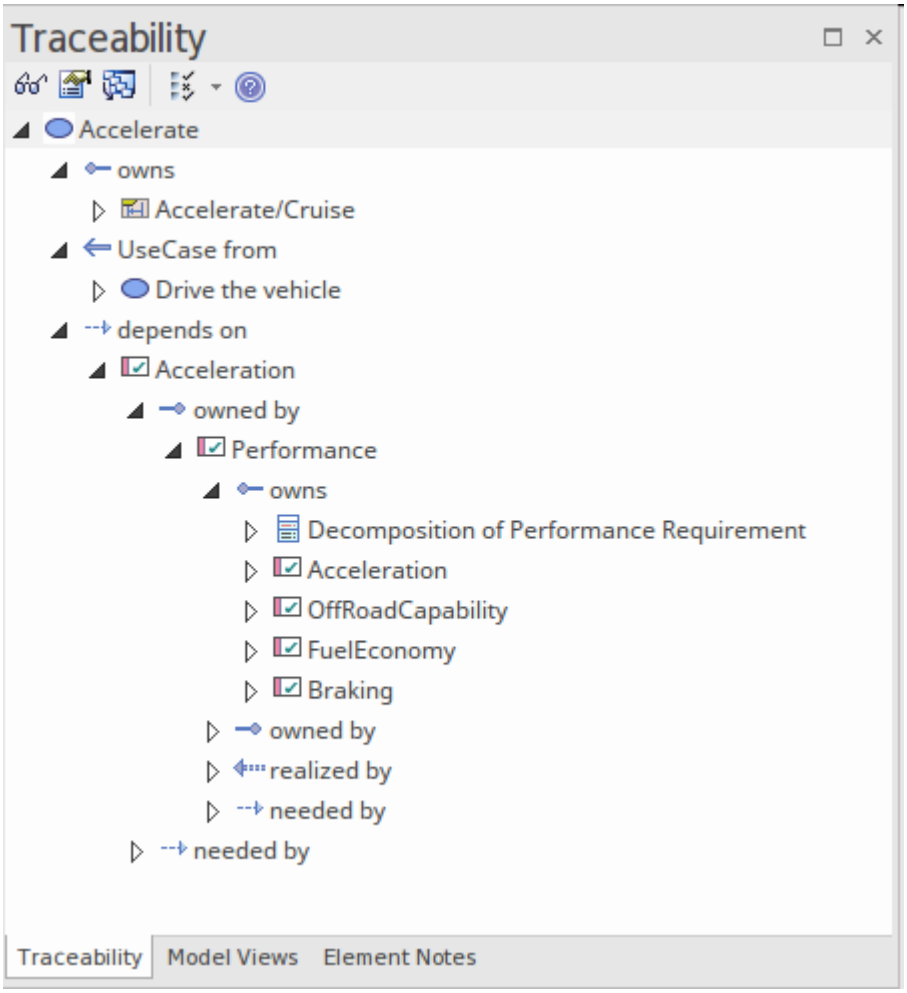


Behavioral Models

A systems engineer can describe how the structural elements of a model behave using a series of diagrams collectively known as the Behavioral model. Structural elements exhibit behavior in a running system, and a number of the structural elements themselves have behavioral features such as operations. The system modeling language specification classifies a number of diagrams as behavior. They are all used to represent different aspects of a system's behavior, from the Use Case that describes behavior that is valuable to a user, to a Sequence diagram that articulates how elements interact.

Use Case Diagram

Use Cases and Actors are high-level representations of a system's behavior from the user's point of view. An engineer models the value that a user performing a role with respect to the system derives from the system behavior. A Use Case will typically be traced to other elements such as Requirements and structural elements such as Blocks.

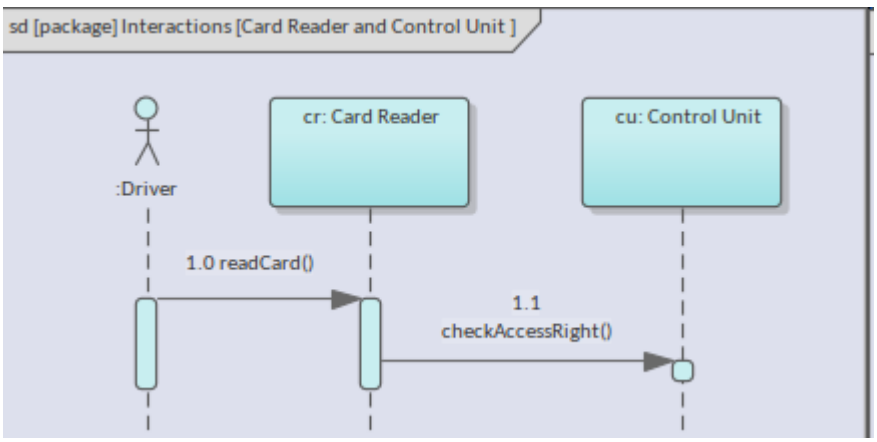


Activity Diagram

Activity diagrams are flow-based models that describe a system's behavior by articulating the flow of items, including information and physical items that act as inputs and outputs to activities and actions as the system performs work.

Sequence Diagram

A modeler uses the Sequence diagram to describe the way messages flow between parts and properties of Blocks. The messages are sequenced and are typically implemented by behaviors such as operations owned by the Block.



StateMachine Diagram

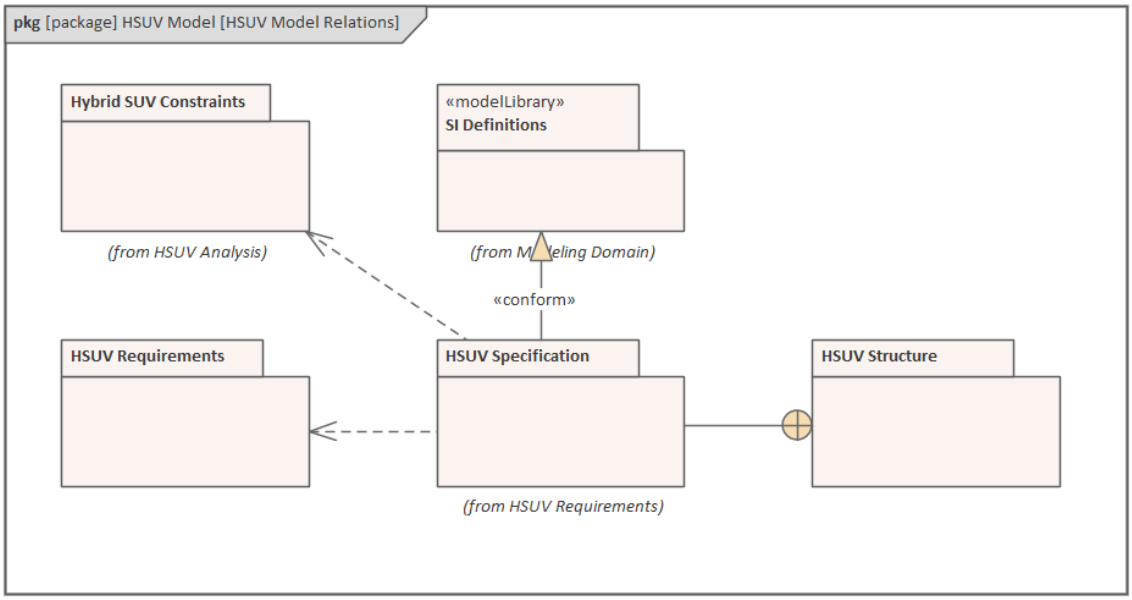
A system engineer uses StateMachine diagrams to describe how structural elements such as Blocks behave in response to events that fire and how the states that a Block exhibits a transition from one State to another.

Structural Models

A systems engineer can describe the structure of a system using a series of diagrams that together form the structural model. These diagrams and the elements and relationships they contain, define the components of the system that are bought to life by the behavioral models. The system modeling language specification classifies a number of diagrams as structural. They are all used to represent different aspects of a system's structure at a logical and physical level, from the Packages that organize the model to Parametric diagrams that define equations and their input and output parameters.

Package Diagram

A complex system must be organized to ensure both humans and other systems can understand, digest, and locate items of interest in the model. Packages that appear in the Browser window can also be placed onto diagrams and are the primary element used to structure the model.



Block Definition Diagram

Blocks are the fundamental structural element in a system's representation; they contain features, exhibit behavior, change states, and interact with other Blocks to produce the behavioral contracts of the system.

Internal Block Diagram

The usage of Blocks and their parts is described on an Internal Block diagram using Parts, Ports, Interfaces and relationships, including flows that describe items that pass between interconnected Blocks.

Parametric Diagram

A property's constraints are defined using Parametric diagrams that model engineering and mathematical equations and their input and output parameters.

Defense and Commercial Architecture Models

A number of frameworks have been used to model large systems or systems-of-systems in defense organizations and large commercial or industrial organizations. These frameworks are based on modeling languages such as the Unified Modeling Language (UML), the Systems Modeling Language (SysML), and Service-Oriented Architecture standards. The frameworks have evolved over several decades as defense, and commercial systems and projects have become larger and more complex. For example, DoDAF and MODAF have been combined to form the basis for the Unified Profile for DoDAF/MODAF (UPDM), and this, in turn, has evolved into Unified Architecture Framework (UAF). Enterprise Architect has rich support for both UPDM and UAF, and systems engineers can create robust, expressive, and compliant defense and commercial models that provide views of complex systems or systems of systems.

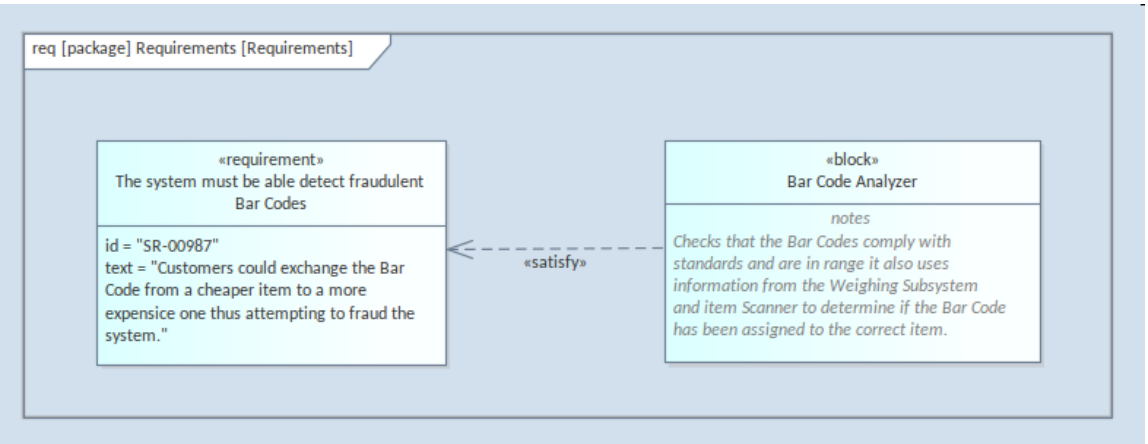
Requirements Models

Requirement Engineering is one of the most important disciplines in the system lifecycle. When done well, it will set the foundation for a successful project or program of work, ultimately ensuring that engineering teams deliver great value to the users and other stakeholders. Enterprise Architect is a sophisticated and intuitive platform for developing and managing Requirements gleaned from modeling stakeholder statements, business cases, business drivers, and capabilities to define detailed Functional and Non-functional Requirements. The engineer can prioritize, trace, and track requirements and record changes, baseline, version, and record audits of changes. Engineers, managers, consultants, and customers can work together in a collaborative platform with role-based Security, Discussions, a team Library, Model Mail, and a range of other tools to encourage best practice and productivity, either directly on the local system or through Pro Cloud Services.

Requirement	Priority	SysML1.4::text	Stereotype	Status	Difficulty
Illumination	Medium	The system must use strip lighting for illuminating the boom.	requirement	Proposed	Low
Minimize Power Utilization of Boom Gate	Low (Critical / High / Medium / Low)	The system must minimize the power used by all of its components	requirement	Proposed	High
Operational Visibility	High	The system must ensure any barrier is visible in all operating conditions including weather events such as fog and low light conditions such as at night.	requirement	Approved	Medium
Fog and Rain Visibility	High	The system must ensure any barrier is visible in any weather conditions including Fog and Rain and there must be enough time in these conditions for a driver to stop at the control unit.	requirement	Validated	High
Low Light Visibility	Critical	The system must ensure any barrier is visible in low light conditions including night and shadows and there must be enough time in these conditions for a driver to stop at the control unit.	requirement	Implemented	Low

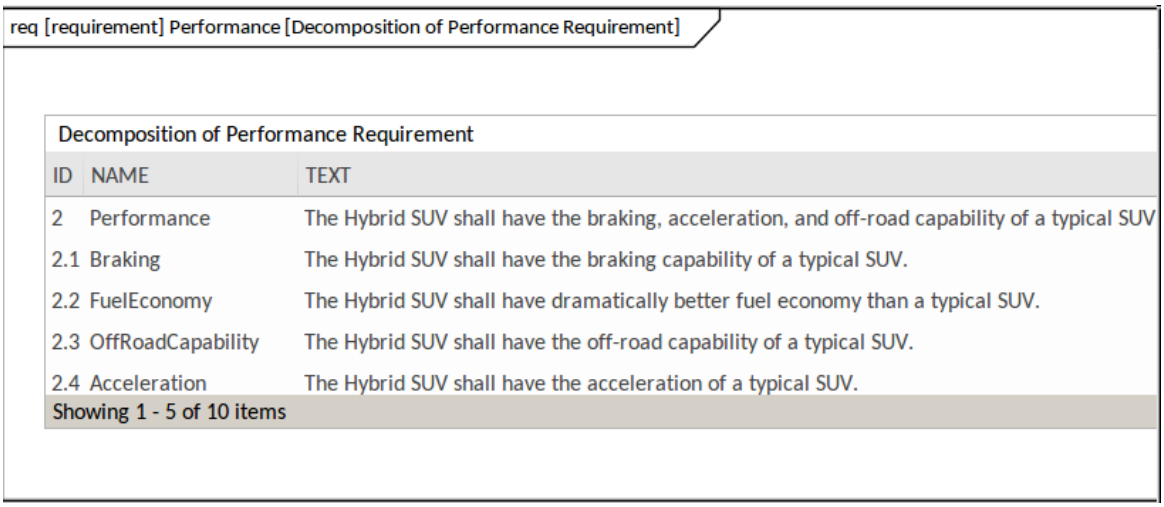
Requirements Diagram

A systems engineer uses a requirements diagram to create and view Requirements and their relationships to other elements, including other Requirements. You can specify requirements at any level, from strategic enterprise or business requirements through stakeholder requirements down to low-level engineering and even software and transition requirements. Requirements properties, including their id and text, can be displayed or suppressed on a diagram; the choice depends on the diagram's purpose and its intended audience.



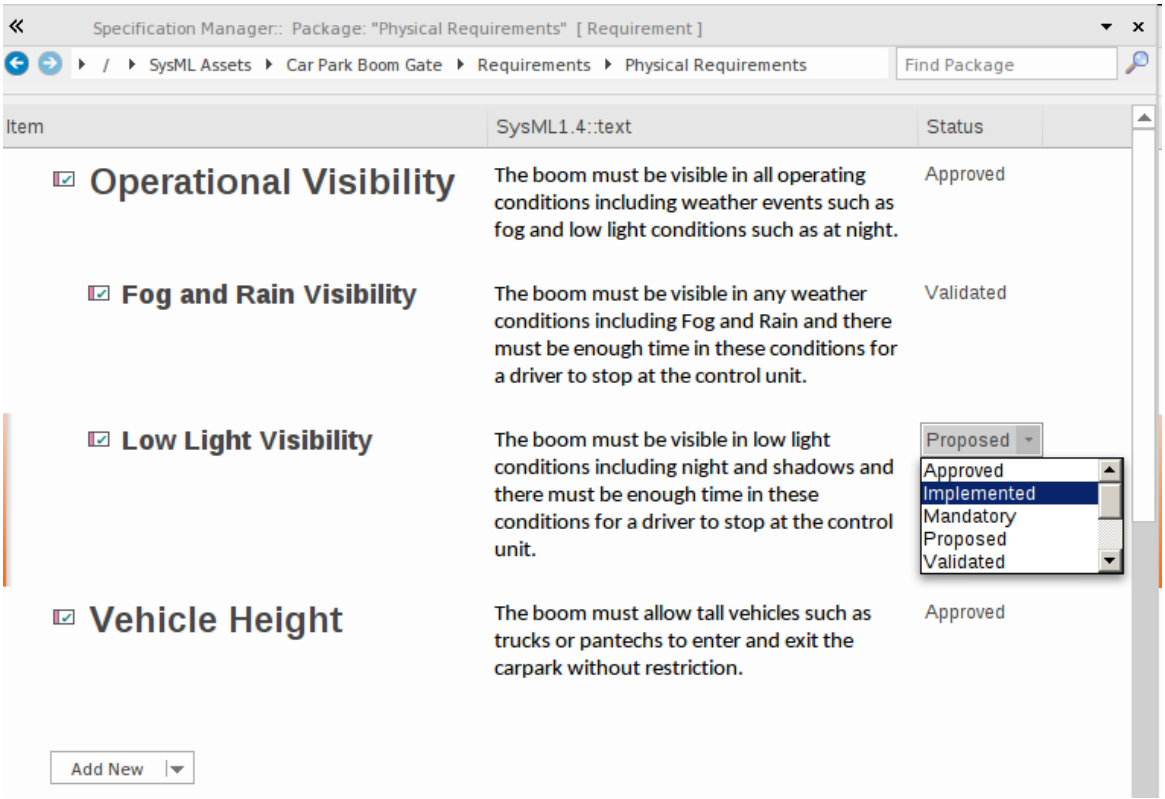
Requirements Discipline

Requirement development includes all the activities and tasks associated with discovering, evaluating, recording, documenting, and validating the requirements for an engineering project or program of work. Requirements are discovered, analyzed, specified, and verified, and Enterprise Architect has a wide range of tools and features to assist the requirement engineer as they develop requirements. The Specification Manager is the centerpiece for requirement development, allowing the Analyst to enter, view, and manage requirements in textual form in a spreadsheet or document-like view.



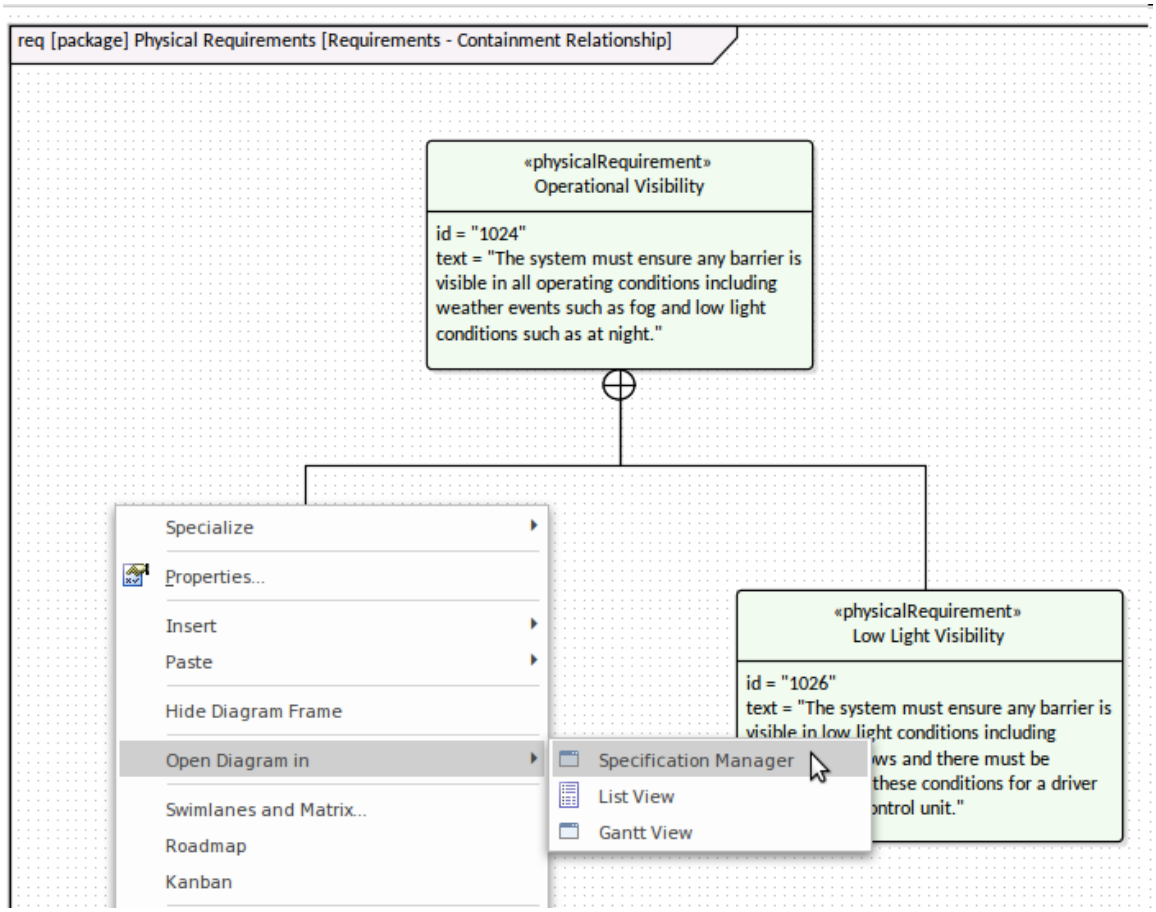
Requirements Tools

The Specification Manager is an easy-to-use tool providing a spreadsheet or word processor view that you can use to manage requirements or any other model element. It is particularly beneficial when working with requirements that have descriptive text to describe the requirement in detail. An engineer can create new requirements with names and detailed descriptions, and properties such as Status and Priority can be added or changed from drop-down lists. You can conveniently view and manage existing requirements using other diagrams and windows - and changing them in the Specification Manager will change them in all other places in the repository.



The Specification Manager is the perfect tool for those analysts who are more comfortable working with text rather than diagrams and who are accustomed to working in a Word Processor or Spreadsheet. It has the added advantage that the Requirements are part of a model, and an engineer can trace them to other elements, including Business Drivers, Stakeholders, and Blocks. This image shows that you can specify and manage the Requirement status and other element properties using drop-down lists.

An engineer can open diagrams and Packages containing requirements in many views, including the Specification Manager.

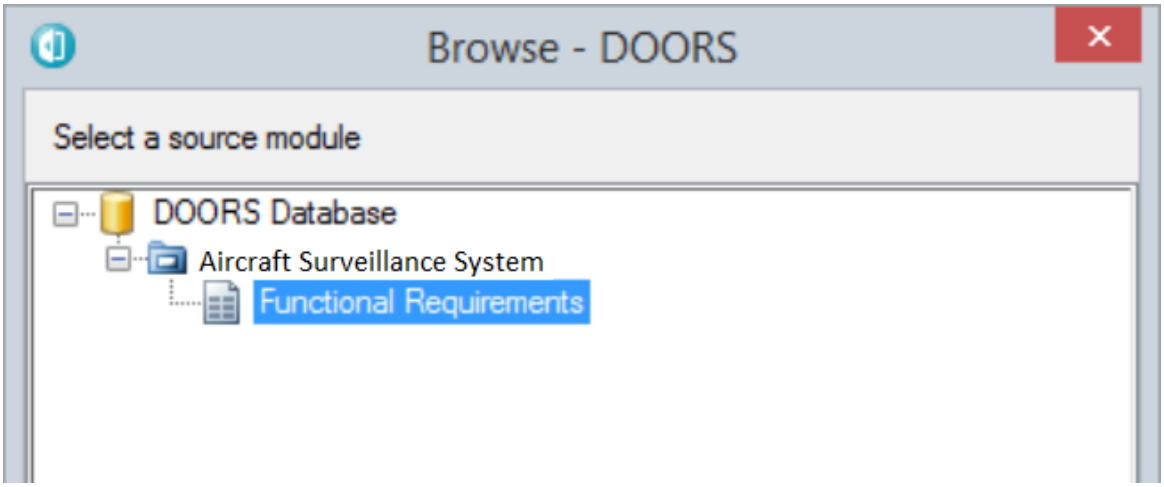


DOORS Integration

The designers of Enterprise Architect understand that customers might have existing or incumbent requirement management tools that they use as part of a corporate or engineering policy. To allow the full scope of modeling and traceability, Enterprise Architect integrates with both the legacy DOORS application and the newer DOORS Next Generation tool, allowing you to view the DOORS requirements inside the application and trace to both up-process and down-process elements these requirements.

DOORS MDG Link (Legacy method)

In the MDG Link for DOORS, you can create a link between Sparx Systems Enterprise Architect and an existing IBM® Rational® DOORS® module, which enables you to exchange requirements data between DOORS and Enterprise Architect. You can also redirect the link to a different module. You can import data from DOORS to Enterprise Architect and export data from Enterprise Architect to DOORS through this link. You can both import requirements from DOORS or export requirements located in the Enterprise Architect repository to DOORS.



DOORS NG Integration

Using the Pro Cloud Server Jazz Plug-in integration you can manage various Rational tools, including the DOORS Next Generation Requirement Management tool. This allows you to push and pull requirements from any configured DOORs project. The tools include:

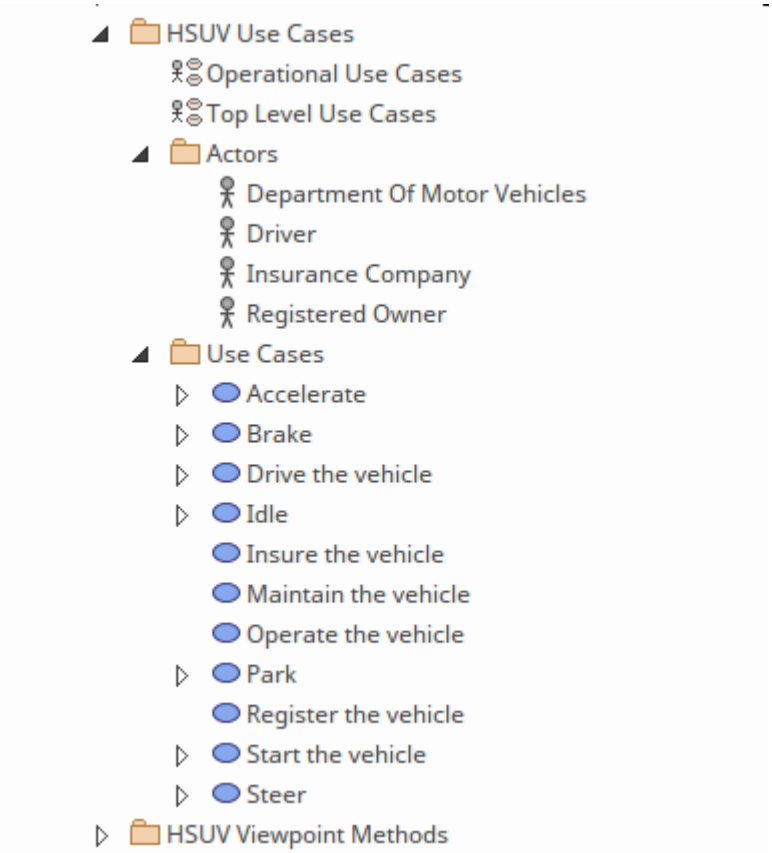
- IBM Rational DOORS Next Generation's Requirement management tool
- Rational Rhapsody Design Management (DM)
- Rational Team Concert Change and Configuration Management (CCM)
- Rational Quality Manager (QM)

Structural Models

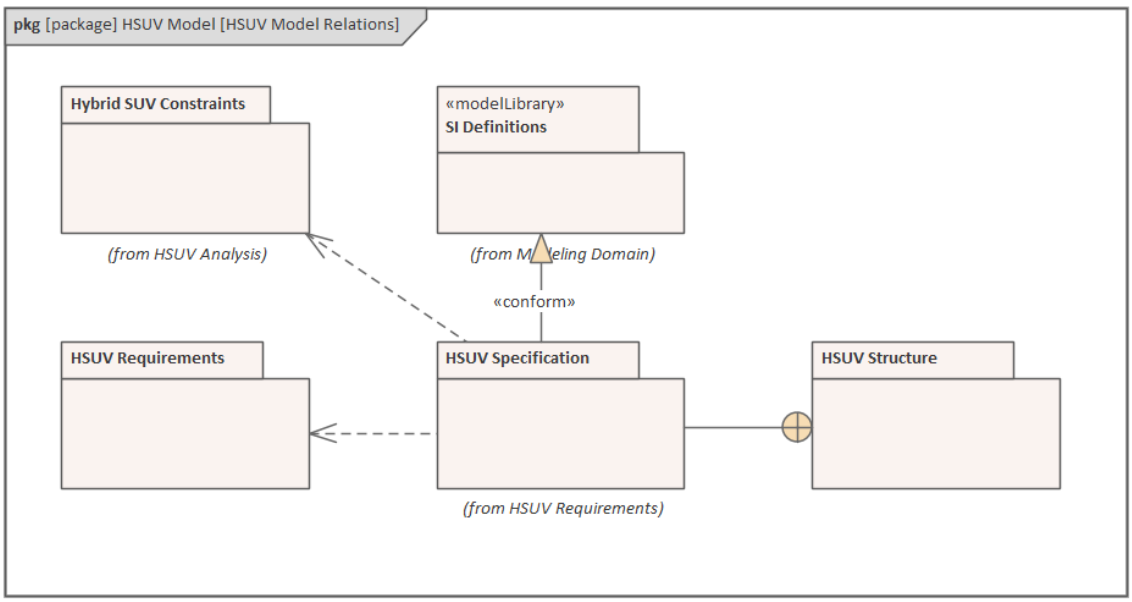
The structural models contain the 'nouns' of the system and define the structures or components of the system. Packages are the primary element for structuring a model or repository and act as containers or namespaces for other elements and their features, including other Packages. The fundamental element of structure is the Block, which can contain both structural and behavioral features and can be used to model any logical or physical aspect of a system. Blocks are typically created and viewed on Block Definition diagrams and also appear on Internal Block diagrams that you use to describe the usage of the Block in a particular context showing the parts that make up the Block. Parametric diagrams are a specialized type of Internal Block diagram used for modeling mathematics and physics equations.

Structure with Packages

The organization of a model is critical to the success of a project or the whole engineering level endeavor. The Package is one of the primary and important elements in the SysML for defining structure. It functions as a container and viewed simply, it is resembles a folder in your favorite file explorer software for your computer. So, in this way, it is firstly a container that groups together other elements, including other Packages but it also has other important functions in Enterprise Architect including for version control, baselining, publications and more.



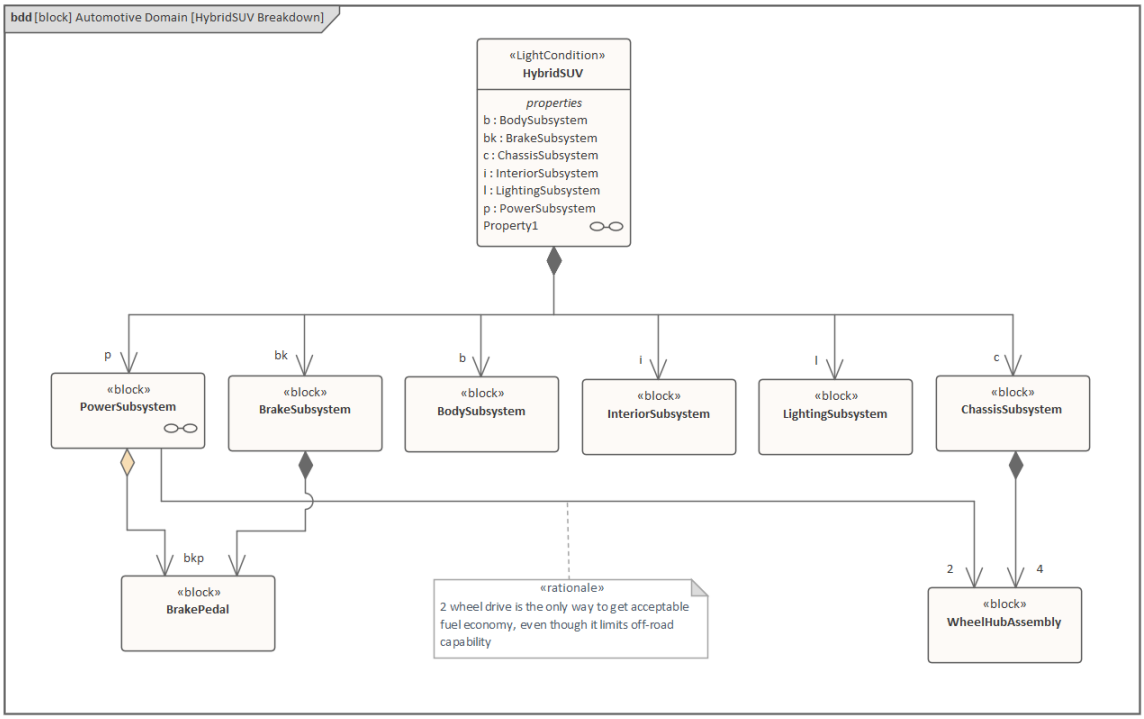
Package diagrams can also be used to visualize the structure of a repository and have the advantage that they can be included in publications or web views of the repository.



Blocks and Constraints

The SysML has similar grammatical categories found in natural languages, with elements that describe structure and other elements that describe behavior. The SysML describes structural things (nouns) using a Block. When engineers create diagrams, they will often use a mixture of behavior or structure elements, describing a particular aspect of a system - bringing to light some aspect of the modeled system.

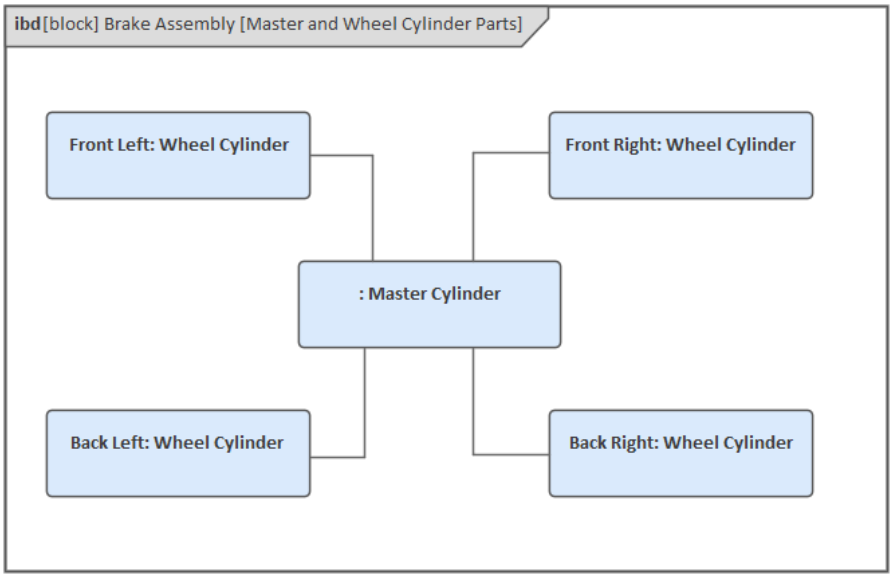
The Block is the fundamental unit of system structure and is used to describe an entire system, a subsystem, a component, an item that flows through a system, a constraint, or entities that reside outside a system. Similar to our natural languages, a Block can represent something abstract, logical, or physical. This is an important concept, and the SysML writers and readers must be clear about the intention of the representation. For example, in a logical architecture, there are typically Blocks representing conceptual ideas or designs that physical and tangible components might realize at the time of detailed design and construction. A systems architect might define a Block called Collision Detection Subsystem that is an expression of a logical system component that could at the detailed design phase, be in part, realized by a set of radar and laser transmitters, detectors and cameras.



Parts and Block Usage

Blocks are classifiers and describe the characteristics of a set of elements that represent how the Block is used in a context. When the Block has attributes (value properties) defined, these are given specific values in the Block instances. Effectively, each Block instance has an identity and typically would have different values assigned that define the Block's state. Enterprise Architect allows these values to be specified using a Set Run State option available from the context menu. Block instances are properties or parts. Thus an engineer working in an automotive domain could define aspects of a vehicle's braking system showing blocks representing a master cylinder's relationship to a wheel cylinder defining a multiplicity of 3..4. The engineer would place instances of these blocks on Internal Block diagrams to express how the parts work together to carry out the behavioral contracts of the system.

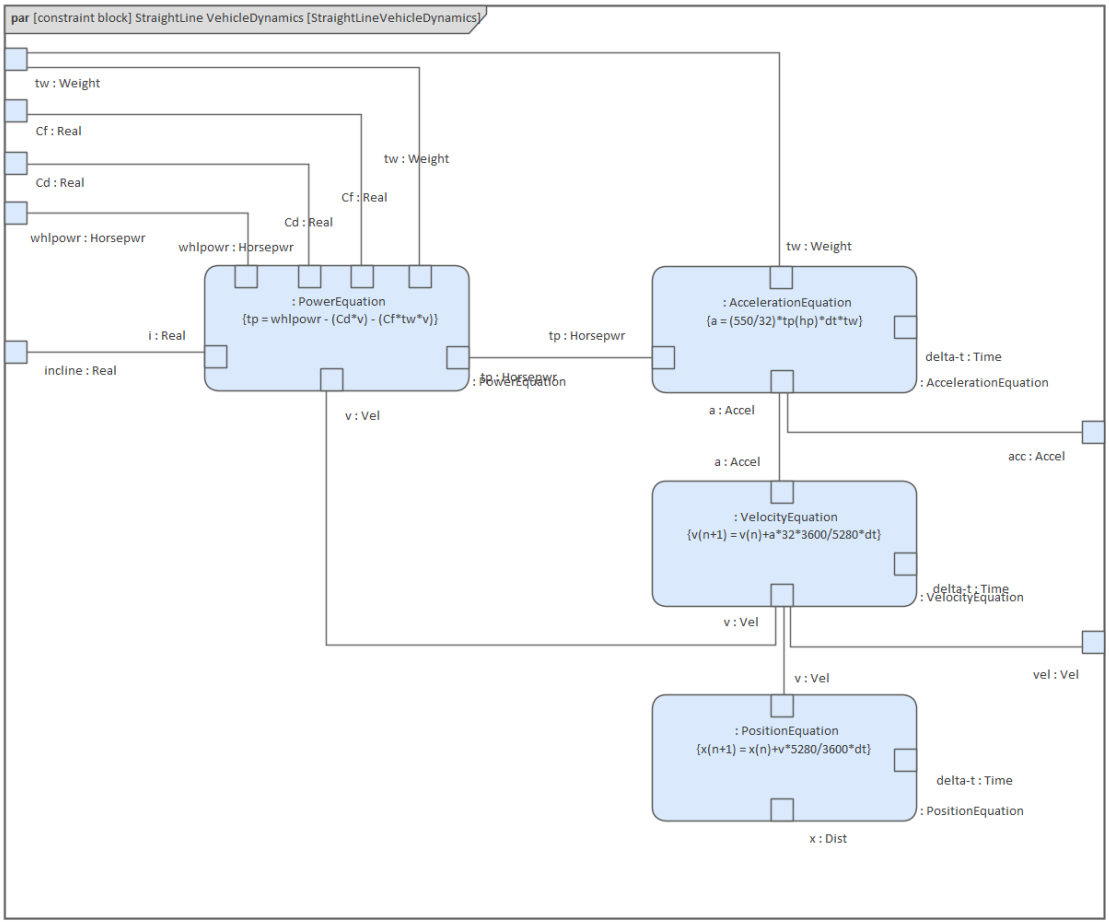
The engineer has named each of the wheel cylinder parts (Front Left, Front Right, Back Left, Back Right) as these need to be identified with respect to their location in the vehicle, but has decided not to name the master cylinder as no further qualification is required.



Parametrics and Equations

Systems engineering models created in Enterprise Architect provide a valuable tool for analysis, design, architecture, testing, and visualization. Systems Engineers are charged with finding solutions to problems and opportunities and using models to visualize the system's simplifications under consideration and the system's operation context or environment. This includes predicting how a system will behave in a given context, balancing competing requirements and design considerations in the form of stakeholder negotiations and trade-off analysis. Parametric diagrams are a powerful tool that can assist the engineer in addressing these concerns in a model and pre-emptively to represent how a system is likely to behave.

Constraints can be modeled on a block definition diagram and then Parametric diagrams are used to show how theses ConstraintBlocks are used in a particular context, being represented on the diagram as ConstraintProperties. We can visualize how the total power parameter is calculated, connecting the Power Equation and the equivalent parameter on the Acceleration Equation. Connections can be seen between the Position Equation and the Velocity Equation, ultimately connected back to the Acceleration Equation.


par [constraint block] StraightLine VehicleDynamics [StraightLineVehicleDynamics]
tw : Weight
Cf : Real
Cd : Real
whlpowr : Horsepwr
tw : Weight
Cf : Real
Cd : Real
whlpowr : Horsepwr
: PowerEquation
{tp = whlpowr - (Cd*v) - (Cf*tw*v)}
i : Real
incline : Real
v : Vel
tp : Horsepwr
: PowerEquation
tw : Weight
: AccelerationEquation
{a = (550/32)*tp(hp)*dt*tw}
delta-t : Time
: AccelerationEquation
a : Accel
acc : Accel
a : Accel
: VelocityEquation
{v(n+1) = v(n)+a*32*3600/5280*dt}
delta-t : Time
: VelocityEquation
v : Vel
vel : Vel
v : Vel
: PositionEquation
{x(n+1) = x(n)+v*5280/3600*dt}
delta-t : Time
: PositionEquation
x : Dist

Behavioral Models

The behavioral models contain the 'verbs' of the system and define how the system behaves from several different viewpoints.

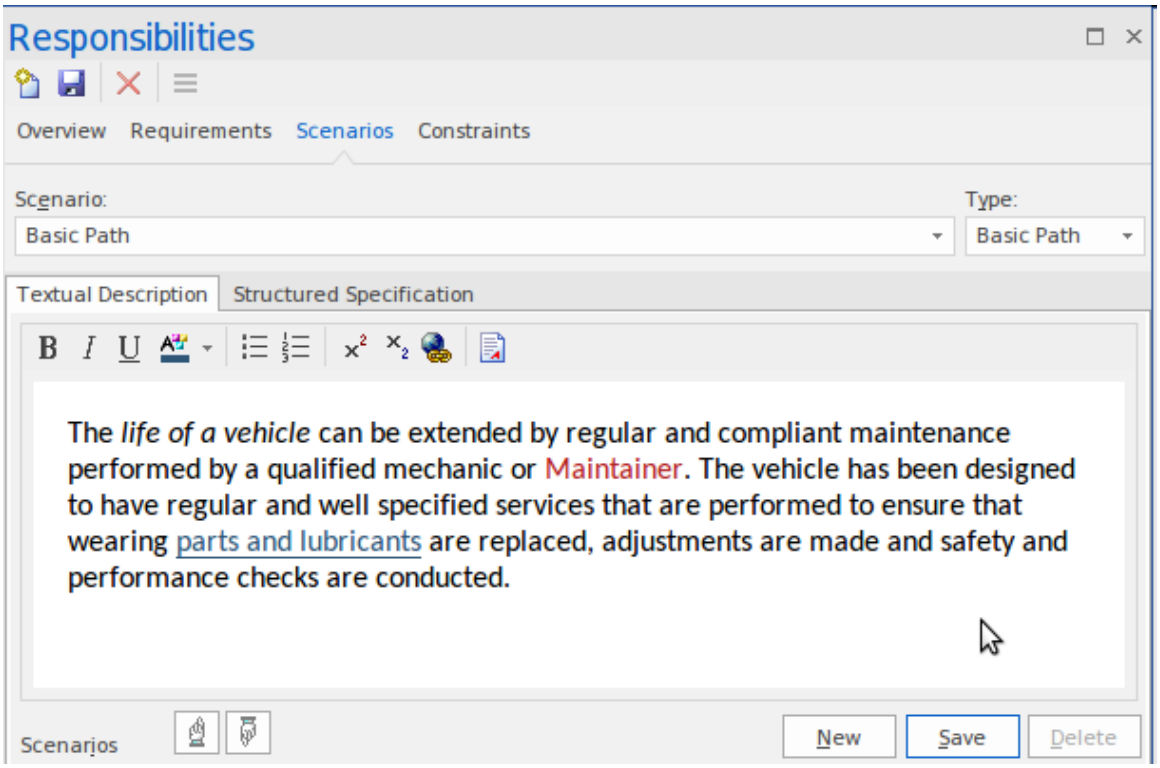
The Behavior diagrams communicate the behavior of the system and demonstrate how the parts of a system work together to satisfy behavioral requirements. Behavioral models have a range of purposes. The engineer must understand what part of the system's behavior they are modeling and then choose the appropriate tool feature and language construct to model this behavior. Systems Engineers use SysML diagrams to model these behavioral characteristics.:

- Use Case diagrams - used to narrow a system's scope and express the users' goals as a value proposition.
- Activity diagrams - used to define the ordered set of actions that carry the work of the system.
- Sequence diagram - used to show how system components or parts interact to produce an outcome.
- State Machine diagram - used to define the discrete states of a system or its parts during its lifetimes.

Enterprise Architect has a range of productivity tools that the systems engineer can use while working with behavioral models, including the Scenario Builder, State Machine Tables, Simulation engine, and many more.

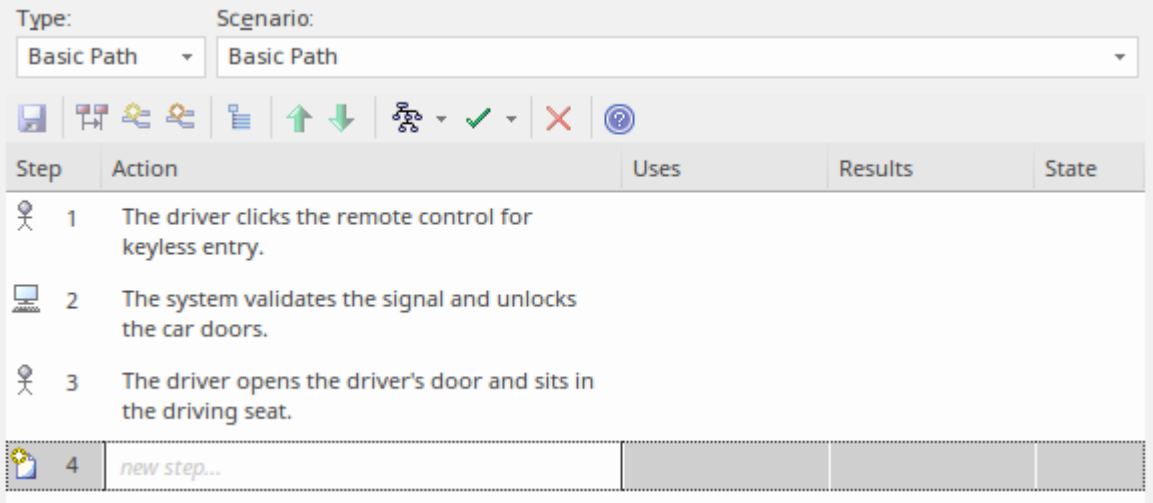
Use Cases and User Goals

The Use Case model describes the value or goals that users (human and system) derive from interacting with the system. A brief description summarizes this value for each scenario, including the all-important basic (sunny day) scenario.

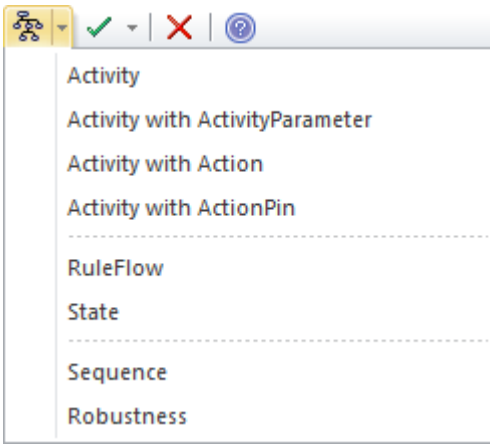


The Use Case technique is fundamentally straightforward and was devised to ensure that functional requirements were written from the User's perspective. This standpoint helped to ensure that deployed systems would be fit for purpose and be accepted by the diverse community of users. There is, however, a vast amount of conflicting literature and an equally large number of styles for defining Use Cases. This situation has led to confusion and uncertainty and has tended to attenuate the value that can be derived from this simple technique.

Enterprise Architect provides a solution to this by including a purpose-built Scenario Editor that the engineer uses to create detailed descriptions of Use Cases, including alternate and exception paths listing the steps performed by the User and the system.



The tool provides a helpful way to generate behavior diagrams such as Activity, Sequence and StateMachine diagrams directly from the scenarios and their steps. These can be synchronized as changes are made to the sequence of steps or to the branch and merge points for alternate and exception scenarios.



Activities and Behavioral Flows

The Activity diagram is an expressive diagram that systems engineers use to show the sequence of actions that describe the behavior of a Block or other structural element. The Actions are sequenced using control flows and can contain input and output Pins that act as buffers for items that flow from one Action to another (or from Control or buffer Nodes). The work carried out by the Actions either consumes or produces these items. The items can be either material, energy, or information, depending on the system and the Activity being described.

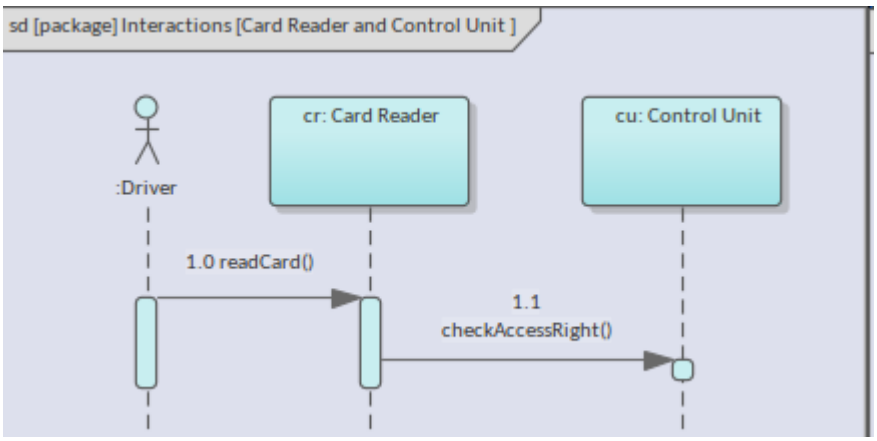
Actions are the behavioral atoms that are connected to describe the behavior of an Activity, Sub-system, system, or one of its parts. Effectively an Activity is made up of a set of actions that work together to convert items (tokens) that are input into the Activity to items (tokens) that are output by the Activity. The first Action in a sequence will receive inputs from one of the owning Activity's Input Parameter Nodes. The last Action in the sequence will place the output onto one of the Activity's Output Parameter Nodes. The Actions themselves have input and output devices called Pins - an Action will receive tokens on its Input Pins, perform its work and place the resulting tokens on its Output Pins.

Sequences and Object Interactions

A system is enacted by its parts, working collaboratively to carry out the behavior specified in the behavioral models. Instances of structural elements interact by exchanging messages. These interactions can be specified and visualized

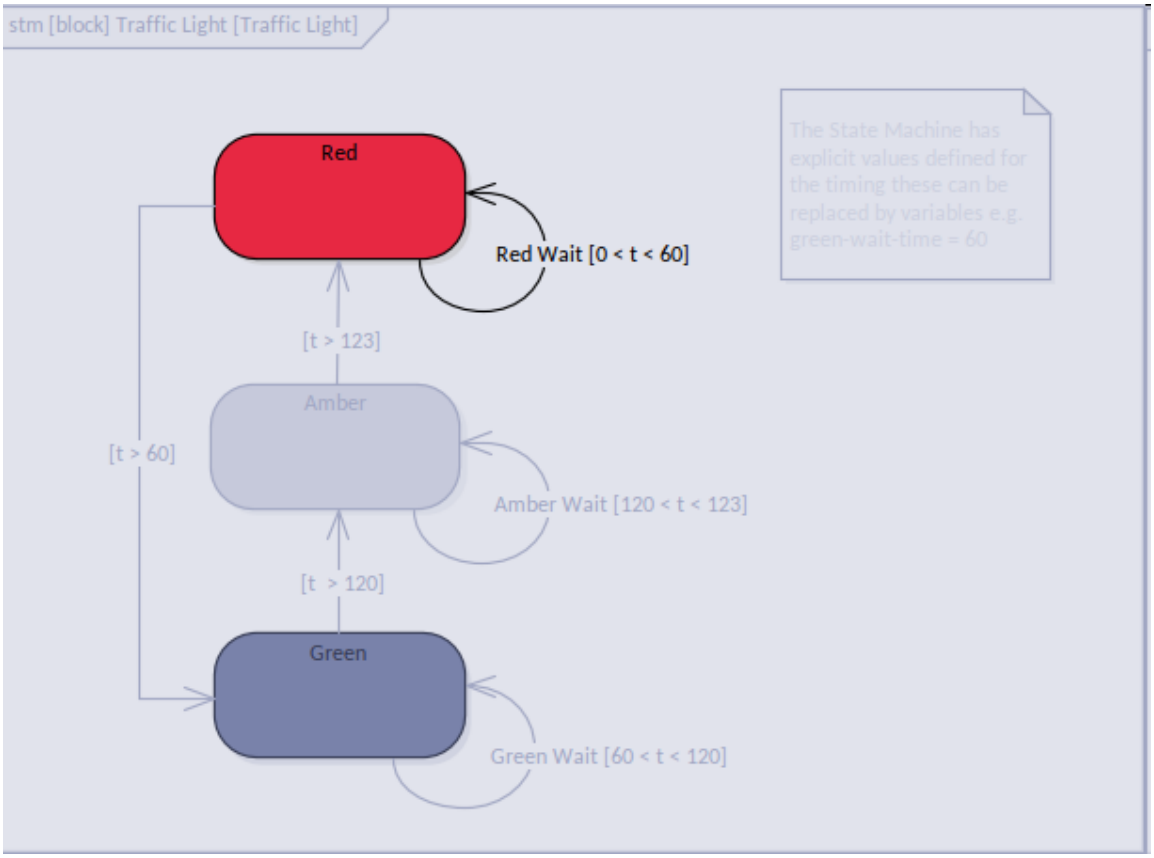
using Sequence diagrams that provide a time-ordered set of messages exchanged between participating instances.

In a Sequence diagram, the Blocks that participate in the interaction have a lifetime that is represented by a dashed line, emanating from the base of the element and continuing vertically for the life of the element. Elements can be created or destroyed at any time during the period represented by the Sequence diagram, and the Lifeline therefore represents their existence. Elements that are present at the top of the diagram are created at the beginning of the interaction. A message exchange between a sender and a receiver will originate in one Lifeline (the sender) and end in another (the receiver).

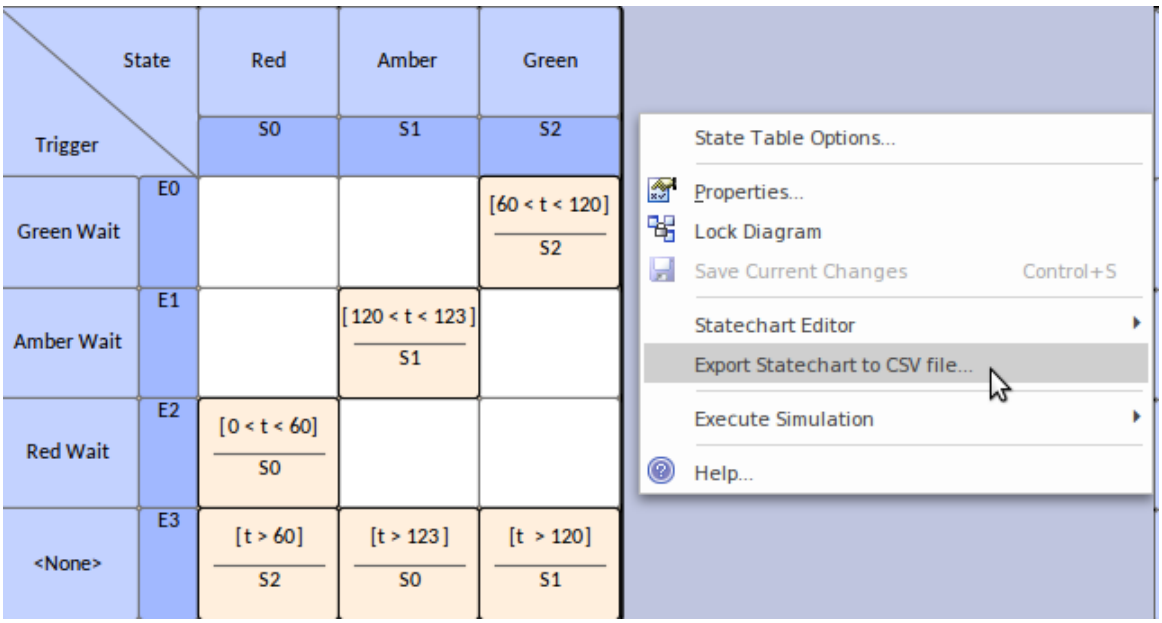


States and Block Lifetimes

The SysML StateMachine is used to describe how structure, modeled with blocks, changes its state in a time-boxed life cycle. Here the engineers' concern is not with the structure of the Block Instance but its behavior, which can, in turn, impact its structure. We are not interested in every single state a 'thing' can exhibit, but rather the significant states. So the critical states for water molecules, for example, could be a solid, liquid, or gas, but we are not ordinarily interested in liquid water at a temperature of 67 degrees Centigrade. If we were looking at a movie reel of an object's lifetime, a StateMachine would pick out the significant frames where significant and relevant changes occurred.

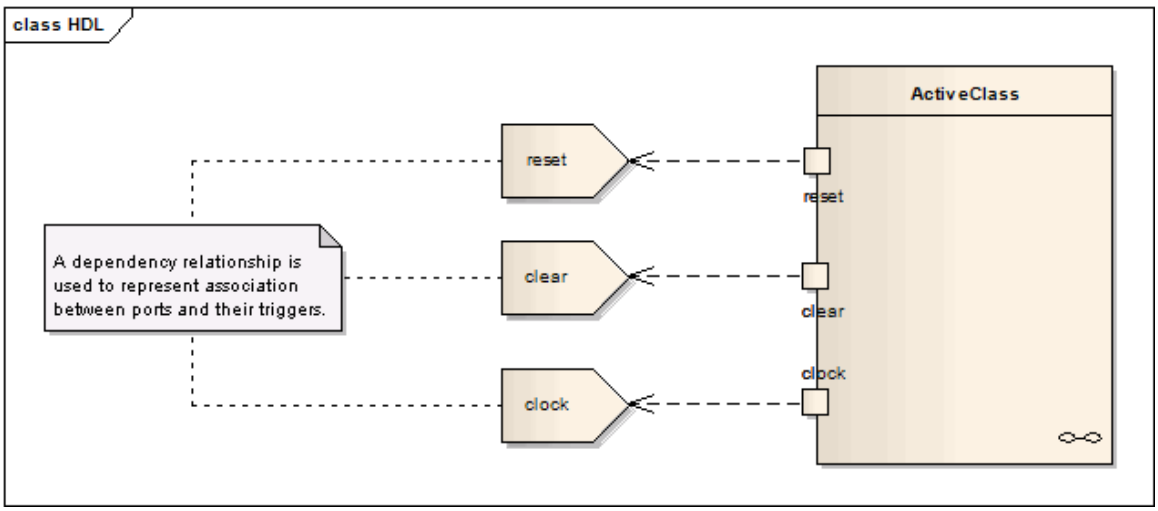


Enterprise Architect also allows an engineer to toggle between diagram and table views of a state machine providing alternative views for engineers and managers who prefer a tabular representation of the state changes. You can also export the tables to a spreadsheet in CSV format for further analysis.



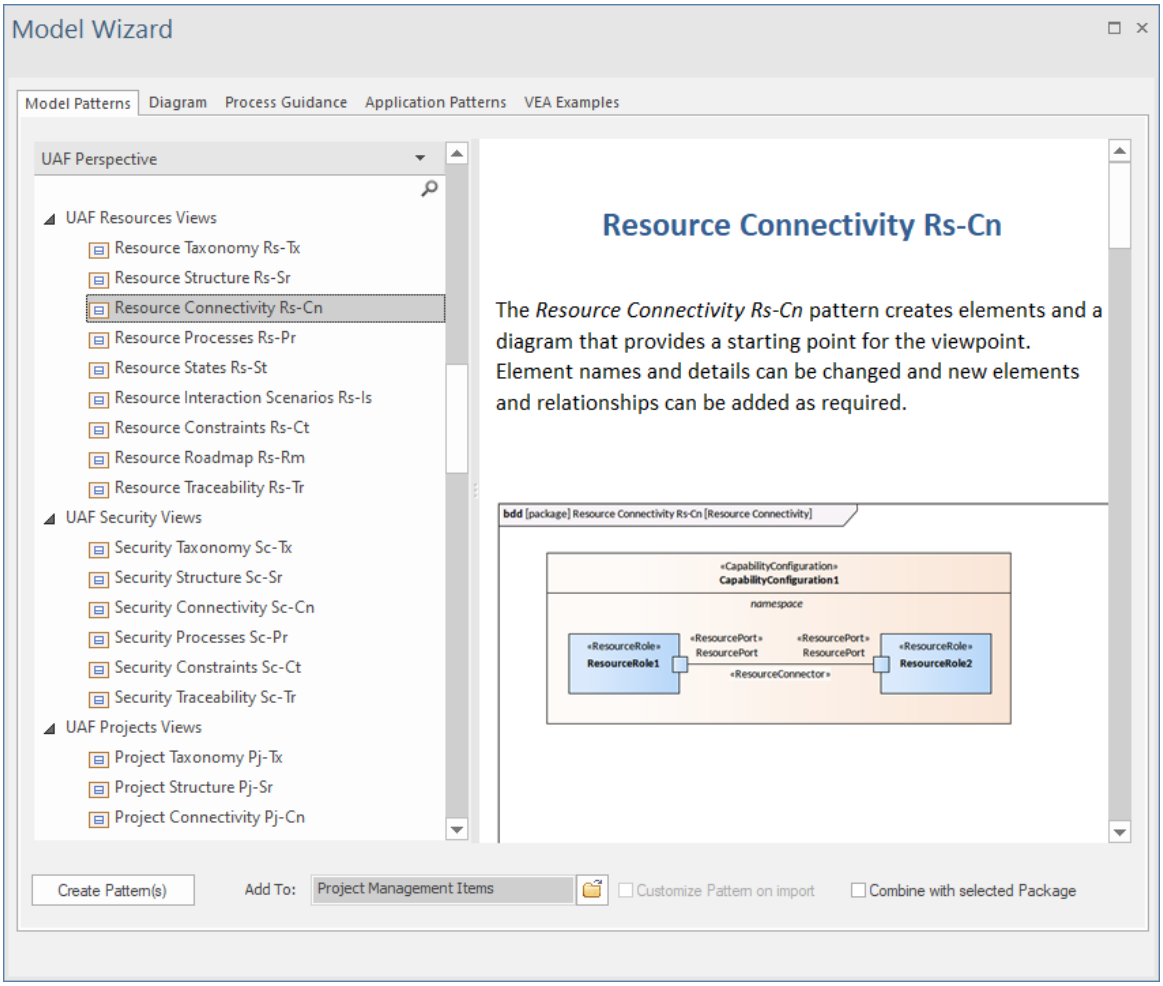
A systems engineer can use state machines to generate executable software code from the model using Executable StateMachines. The code generated is based on its language property. The programming language might be Java, C, C++, C#, or JavaScript; regardless of the language, Enterprise Architect generates the appropriate code immediately ready to build and run.

You can not only generate executable software code, but you can generate Hardware Description Languages and Ada from your model elements, for the chips and circuits in system hardware components.



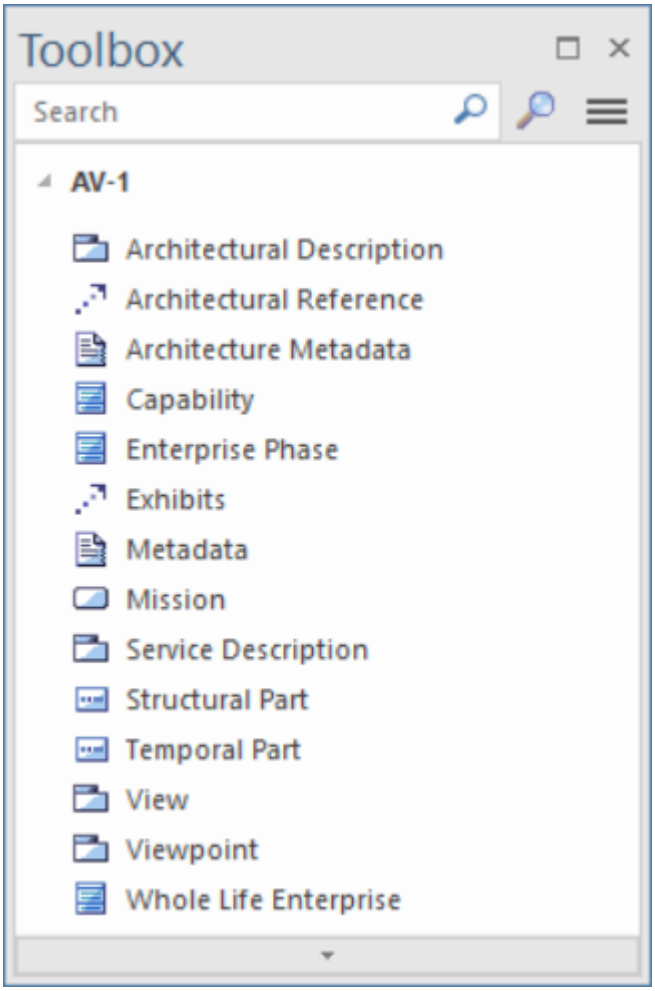
Defense and Commercial Architecture Models

Aerospace and military systems have led the way for model-based system engineering and the use of languages such as the Systems Modeling Language (SysML) and frameworks such as the USA Department of Defense Architecture Framework (DoDAF) and the UK Ministry of Defence Architecture Framework (MODAF) and more recently the Unified Profile for DoDAF/MODAF (UPDM) and the Unified Architecture Framework (UAF). Large-scale commercial organizations have subsequently adopted these languages and frameworks to develop systems-of-system projects. Enterprise Architect has provided tools for creating language-compliant models and has rich support for the frameworks, including pre-built model patterns for both the language models and frameworks.



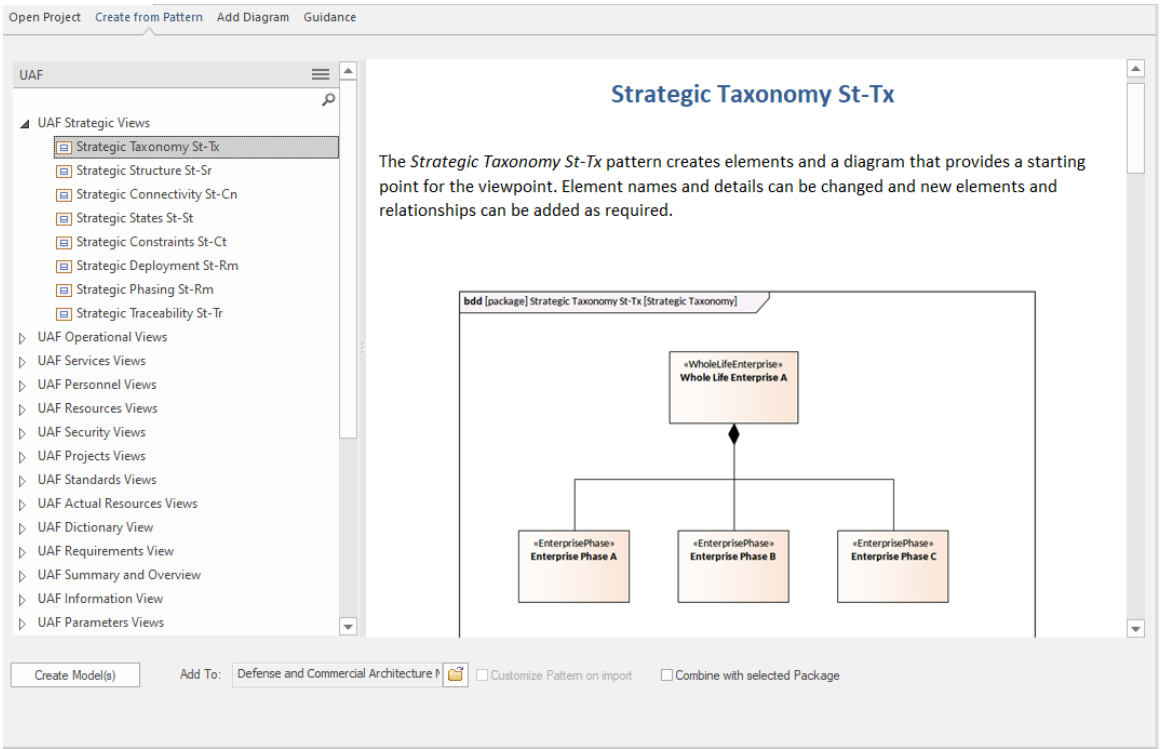
Unified Profile for DoDAF/MODAF (UPDM)

UPDM is an acronym for the Unified Profile for DoDAF/MODAF, which is a unified framework that supports both the USA Department of Defense Architecture Framework (DoDAF) and the UK Ministry of Defence Architecture Framework (MODAF). Historically these were independent frameworks and had heterogeneous architectures and metamodels; UPDM binds them into a single framework. Enterprise Architect has deep support for UPDM. The defense systems engineer can create compliant models using a range of Toolbox pages to create any set of specified viewpoints and views that support the specific needs of stakeholders.



Unified Architecture Framework (UAF)

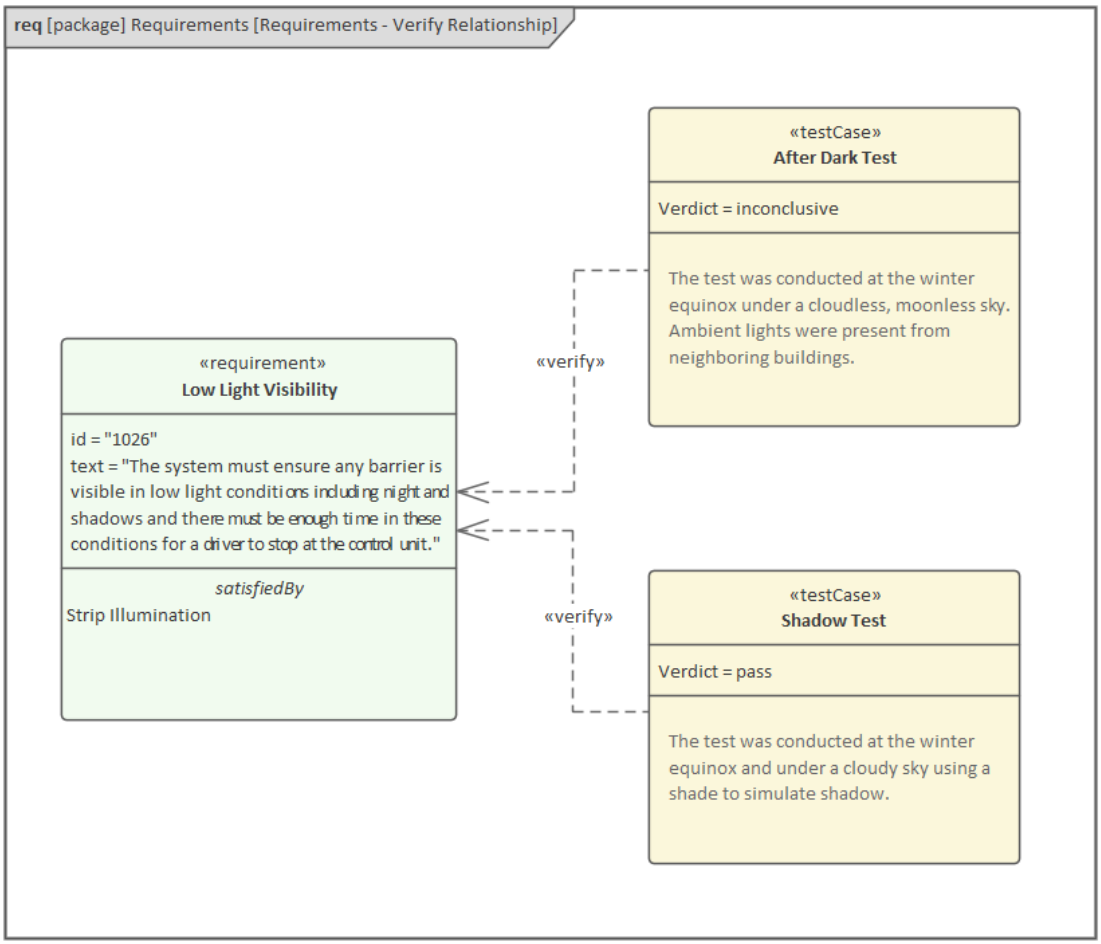
UAF is an acronym for the Unified Architecture Framework, and the framework is based on the Unified Profile for DoDAF and MODAF (UPDM). UAF defines ways of defining and expressing an enterprise architecture that enables stakeholders to focus on specific areas of interest while keeping a high-level view of the entire system. UAF was designed to meet the needs of all stakeholders and allows the creation of specific business, operational, and systems-of-systems integration models of commercial and industrial enterprises as well as the U.S. Department of Defense (DoD), the UK Ministry of Defence (MOD), the North Atlantic Treaty Organization (NATO), and other defense organizations models. Enterprise Architect has a full suite of pre-built patterns with accompanying documentation that can be automatically injected into your models.


Open Project
Create from Pattern
Add Diagram
Guidance
UAF
UAF Strategic Views
Strategic Taxonomy St-Tx
Strategic Structure St-Sr
Strategic Connectivity St-Cn
Strategic States St-St
Strategic Constraints St-Ct
Strategic Deployment St-Rm
Strategic Phasing St-Rm
Strategic Traceability St-Tr
UAF Operational Views
UAF Services Views
UAF Personnel Views
UAF Resources Views
UAF Security Views
UAF Projects Views
UAF Standards Views
UAF Actual Resources Views
UAF Dictionary View
UAF Requirements View
UAF Summary and Overview
UAF Information View
UAF Parameters Views
Strategic Taxonomy St-Tx
The Strategic Taxonomy St-Tx pattern creates elements and a diagram that provides a starting point for the viewpoint. Element names and details can be changed and new elements and relationships can be added as required.
bdd [package] Strategic Taxonomy St-Tx [Strategic Taxonomy]
«WholeLifeEnterprise»
Whole Life Enterprise A
«EnterprisePhase»
Enterprise Phase A
«EnterprisePhase»
Enterprise Phase B
«EnterprisePhase»
Enterprise Phase C
Create Model(s)
Add To:
Defense and Commercial Architecture M
Customize Pattern on import
Combine with selected Package

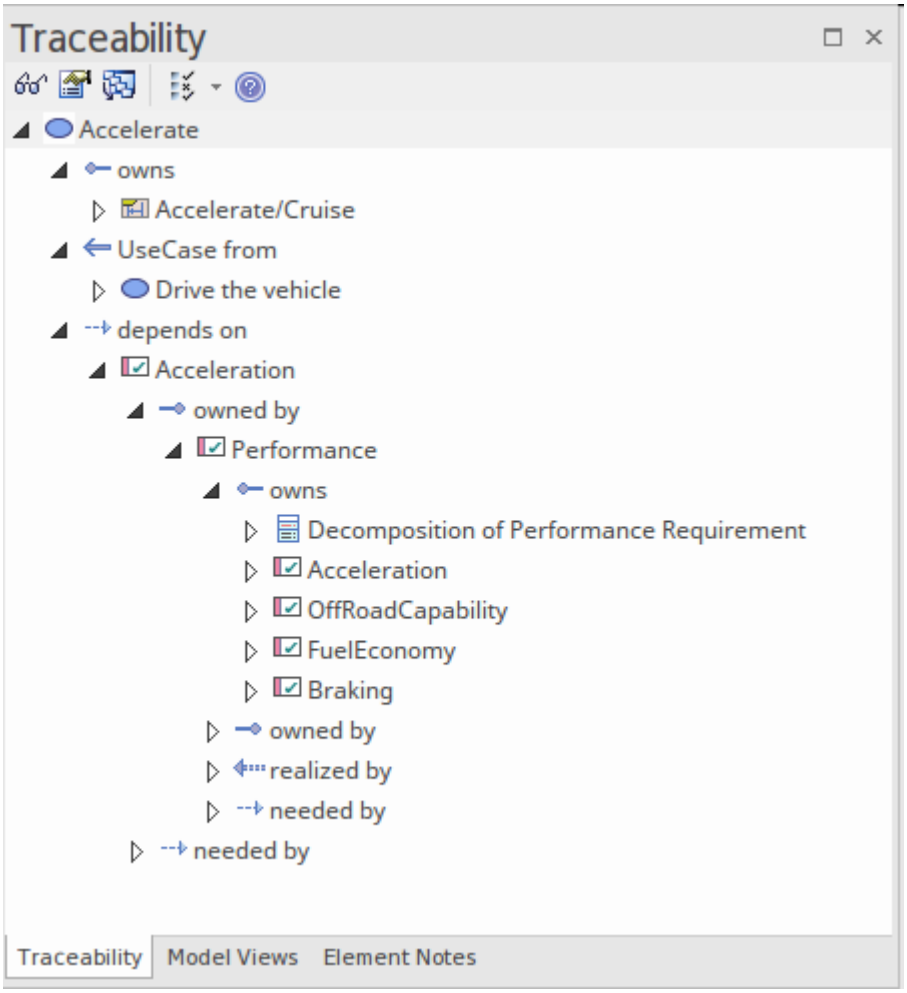
Verification and Validation

Enterprise Architect has a full feature set for verification and validation. Broadly verification is best described as 'has the correct system been built' and validation as 'has the system been built correctly'. The models that the systems engineer creates in Enterprise Architect can be used to verify that the correct system is being built using the traceability features including diagrams the traceability window and the element matrix.

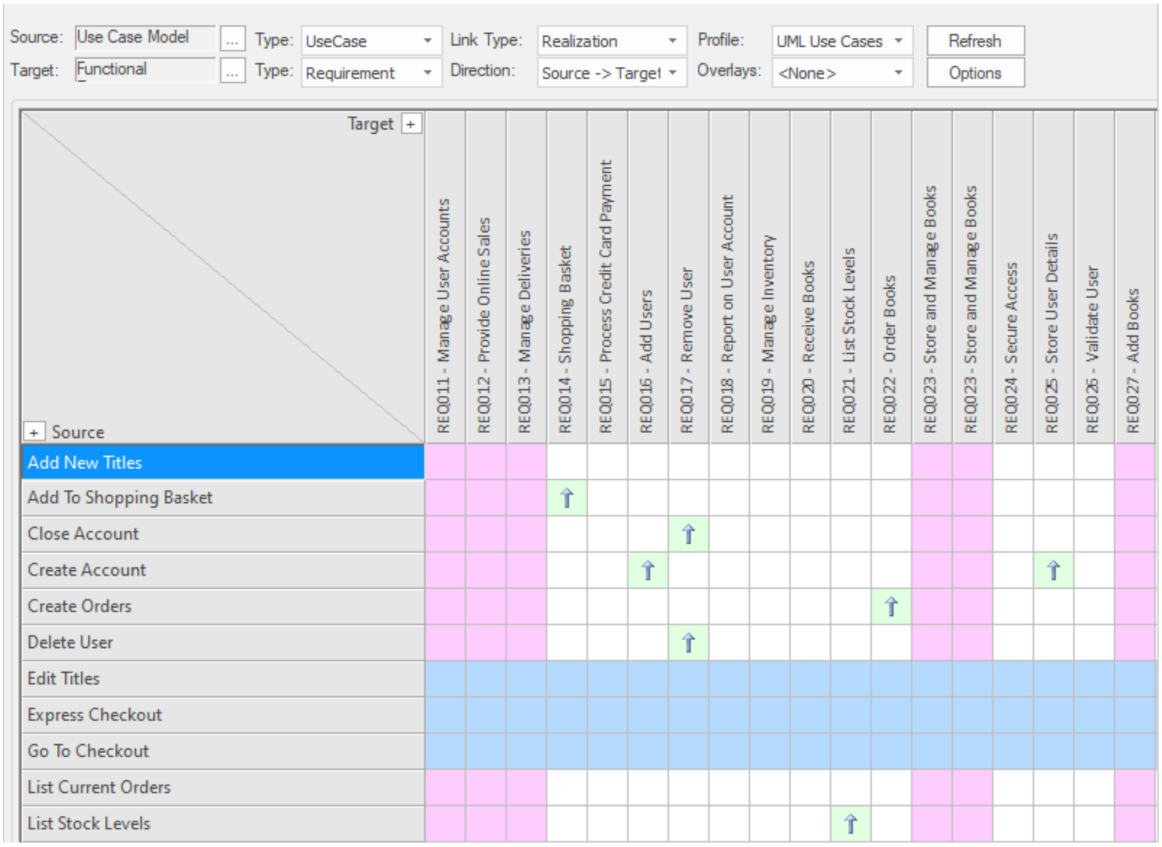
An engineer can use a requirements diagram to express the relationship between Requirements and test cases effectively verifying that the requirements a met by the system.



Modelers can use the traceability window to find and visualize connections between elements. The window is particularly useful for understanding analyzing and verifying that the system's requirements have been implemented by some part of the built system.



A system engineer or project manager can use the matrix window with its tabular view to quickly understand and verify if the system's requirements have been linked to the parts of the system. Colored bands indicate



Enterprise Architect also has a large feature set for specifying tests and recording the results. Enterprise Architect is not only a Systems and Software Modeling environment, it is also a complete Test Management environment. Using Enterprise Architect you can create and manage test scripts for model elements, developing unit, integration, scenario, system, acceptance and inspection tests; these can include test cases generated from xUnit testing and Testpoint Management. Validation and testing teams can also record the test results using the test windows.

You can also import or move tests from other elements, generate them from scenarios, and generate test documentation and reports; you can indicate the presence of tests on an element by displaying test information in a compartment of the element in a diagram, which in turn can be included in documentation and browser based views of the elements.

Simulation and Visualization

One of the great benefits of creating system models is the ability to see the system in motion by running simulations - effectively you to visualize and analyze a system. Enterprise Architect provides a suite of tools to simulate the execution and behavior of the processes that your models define. The tools provide facilities to dynamically simulate a range of types of model including SImulation of Activity and State Machine diagrams, Parametric diagrams, Decision Models and more. Executable State Machines provide rich support for programming language-specific implementations. OpenModelica and MATLAB Simulink can be used to support rapid and robust evaluation of how a SysML model will behave in different circumstances.

Dynamic Simulation

Model Simulation brings your behavioral models to life with instant, real-time behavioral model execution. Coupled with tools to manage triggers, events, guards, effects, breakpoints and Simulation variables, plus the ability to visually track execution at run-time, the Simulator is a multi-featured means of 'watching the wheels turn' and verifying the correctness of your behavioral models. With Simulation you can explore and test the dynamic behavior of models. In the Corporate, Unified and Ultimate Editions, you can also use JavaScript as a run-time execution language for evaluating guards, effects and other scriptable items of behavior.

Extensive support for triggers, trigger sets, nested states, concurrency, dynamic effects and other advanced Simulation capabilities, provides a remarkable environment in which to build interactive and working models that help explore, test and visually trace complex business, software and system behavior. With JavaScript enabled, it is also possible to create embedded COM objects that will do the work of evaluating guards and executing effects - allowing the Simulation to be tied into a much larger set of dependent processes. For example, a COM object evaluating a guard condition on a State Transition might query a locally running process, read and use a set of test data, or even connect to an SOA web service to obtain some current information.

As Enterprise Architect uses a dynamic, script driven Simulation mechanism that analyzes and works with UML constructs directly, there is no need to generate intermediary code or compile simulation 'executables' before running a Simulation. This results in a very rapid and dynamic Simulation environment in which changes can be made and tested quickly. It is even possible to update Simulation variables in real time using the Simulation Console window. This is useful for testing alternative branches and conditions 'on the fly', either at a set Simulation break point or when the Simulation reaches a point of stability (for example, when the Simulation is 'blocked').

In the Professional Edition of Enterprise Architect, you can manually walk through Simulations - although no JavaScript will execute - so all choices are manual decisions. This is useful for testing the flow of a behavioral model and highlighting possible choices and processing paths. In the Corporate, Unified and Ultimate Editions it is possible to:

- Dynamically execute your behavioral models
- Assess guards and effects written in standard JavaScript
- Define and fire triggers into running Simulations
- Define and use sets of triggers to simulate different event sequences
- Auto-fire trigger sets to simulate complex event histories without user intervention
- Update Simulation variables 'on the fly' to change how Simulations proceed
- Create and call COM objects during a Simulation to extend the Simulation's reach and input/output possibilities
- Inspect Simulation variables at run time
- Set a script 'prologue' for defining variables, constants and functions prior to execution
- Use multiple Analyzer Scripts with differing 'prologues' for running the Simulation under a wide range of conditions

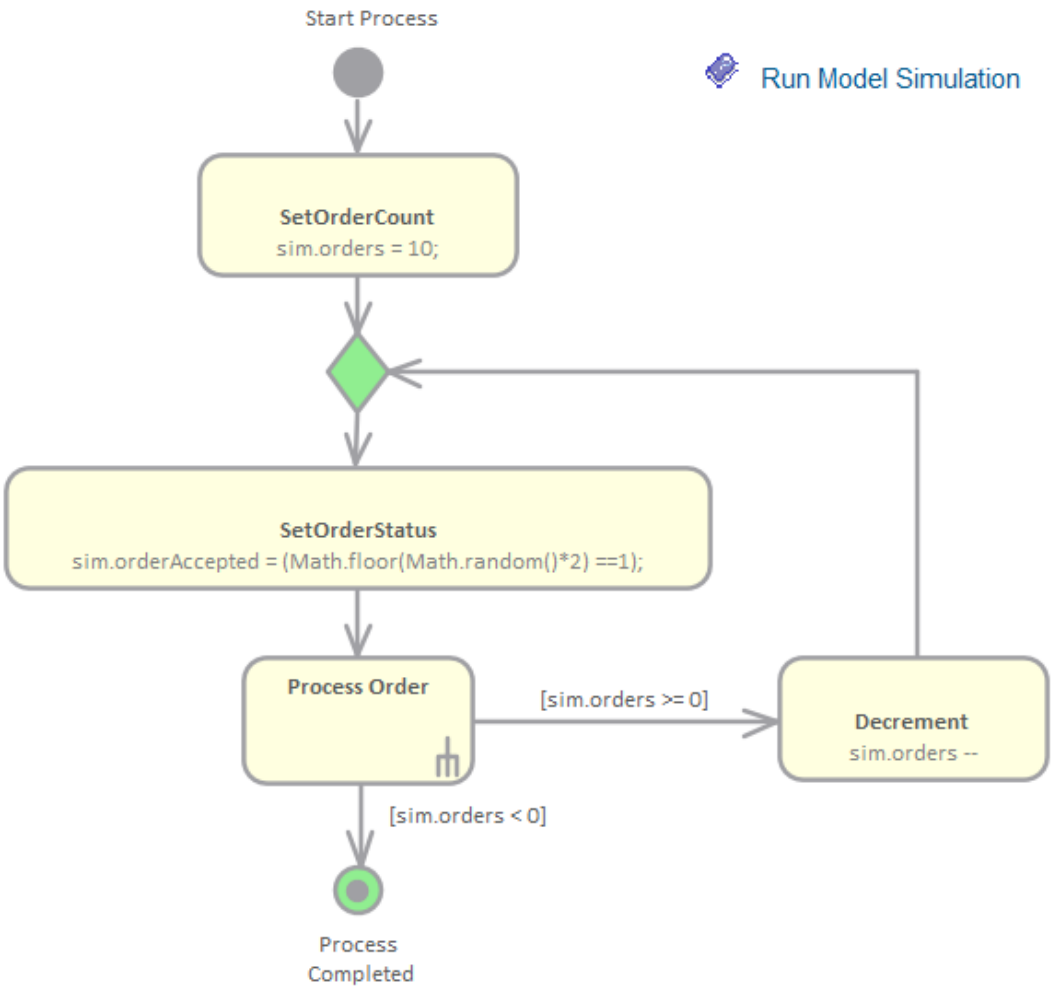
In the Unified and Ultimate Editions it is also possible to simulate BPMN models.

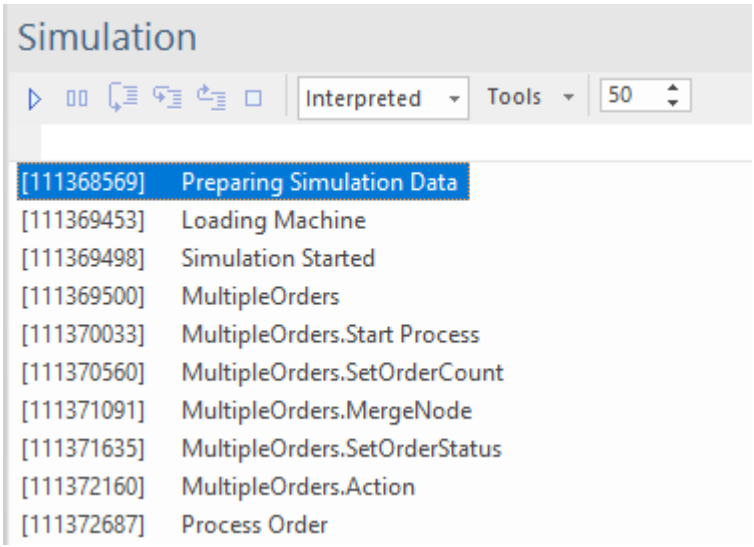
Using the Model Simulator, you can simulate the execution of conceptual model designs containing behavior. When you start a Simulation, the current model Package is analyzed and a dynamic Simulation process is triggered to execute the model.

To get up and running with Simulation, the only steps required are:

- Build a behavioral diagram (State or Activity for manual or dynamic execution, Sequence for manual interaction only)
- Optional: load the 'Simulation Workspace' layout - a fast way of bringing up all the frequently used Simulation windows
- Click on the Simulator Play button

If the diagram contains any external elements (those not in the same Package as the diagram) you will have to create an Import connector from the diagram's Package to the Package containing the external elements. To do this, drag both Packages from the Browser window onto a diagram and then use the Quick Linker arrow to create the connector between them.





Mathematical Simulations

Enterprise Architect provides a wide range of options for introducing advanced mathematical tools and capabilities into your simulations.

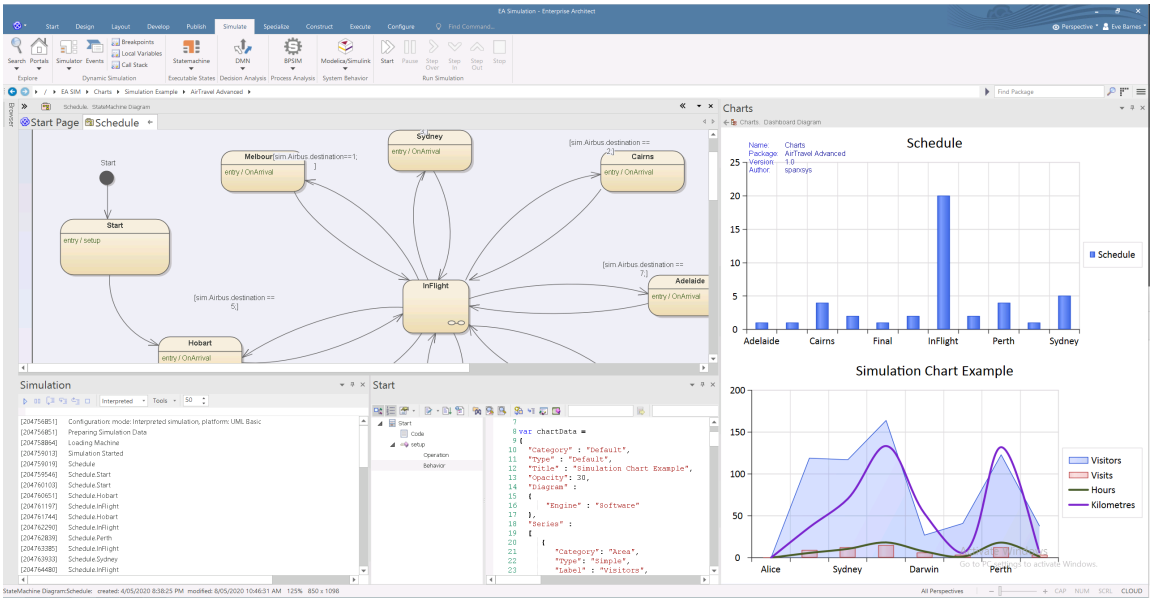
You can bring the power of integrated external tools such as MATLAB into your models through the use of Solver Classes, and can also export your models for execution in other external tools such as MATLAB Simulink, Stateflow and Simscape, or OpenModelica.

Enterprise Architect includes an extensive library of mathematical functions within the JavaScript engine, providing the benefits of a significantly expanded Simulation capability.

Enterprise Architect also provides a wide range of Dynamic Charts; without the need for external tools, you can configure these Charts to extract and plot information from Simulations that have been directly executed inside Enterprise Architect.

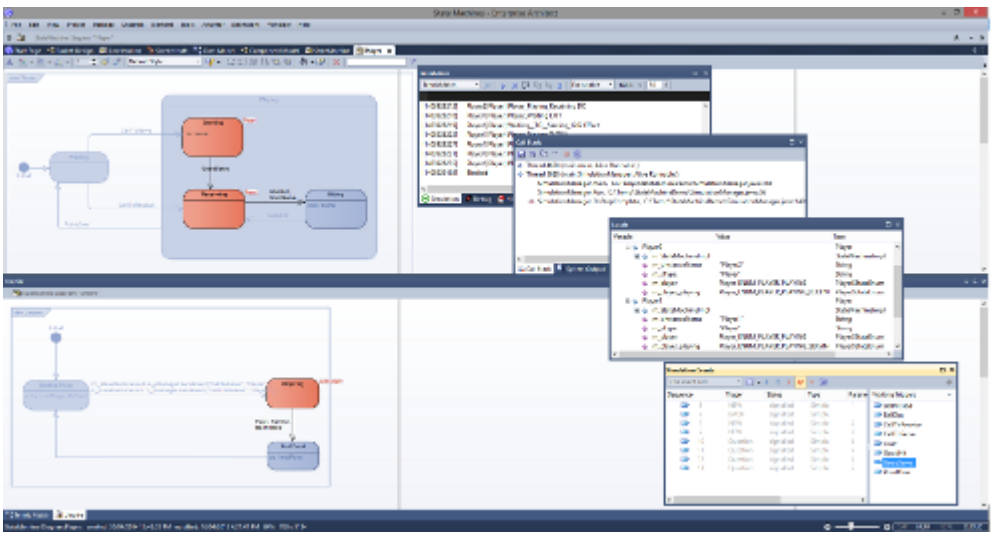
Explore the:

- Solver classes in Enterprise Architect that call MATLAB or Octave to incorporate complex mathematics into your model-based simulations
- Extensive internal Math Library based on the popular Cephes function library
- Integration with the OMG SysPhS standard, enabling you to configure your model for export to common tools
- Support for exporting models to MATLAB Simulink, Simscape and Stateflow; you can create your model in Enterprise Architect and execute it in MATLAB
- Extensive support for Modelica; you can create and configure your model in Enterprise Architect and execute it in Modelica
- Presentation of the results of your modeling and simulation in Chart formats, either within a dedicated graphics presentation tool or through the Dynamic Charting facilities of Enterprise Architect



Executable State Machines

Executable StateMachines provide a means of rapidly generating, executing and simulating complex state models. In contrast to dynamic simulation of State Charts using Enterprise Architect's Simulation engine, Executable StateMachines provide a complete language-specific implementation that can form the behavioral 'engine' for multiple software products on multiple platforms. Visualization of the execution is based on a seamless integration with the Simulation capability. Evolution of the model now presents fewer coding challenges. The code generation, compilation and execution is taken care of by Enterprise Architect. For those having particular requirements, each language is provided with a set of code templates. Templates can be customized by you to tailor the generated code in any ways you see fit.

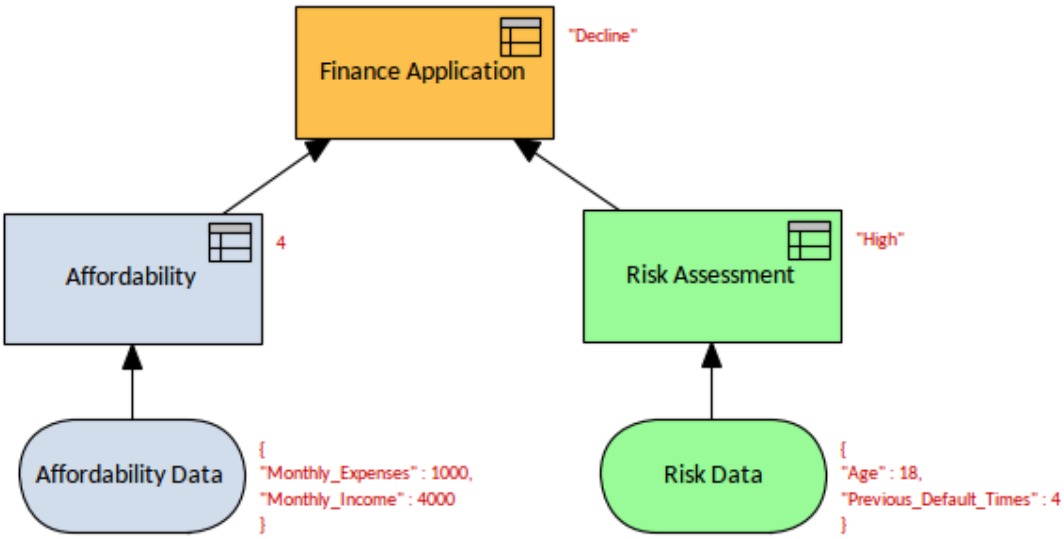


Decision Model Simulation

Organizations face increasingly difficult operating environments with fierce and often unpredictable competition from existing and new market players, changes in government and industry regulations, and upheavals in the social fabric of their customer base. An organization's decisions in this context are critical to its success and its ability to steer a safe path through these uncharted corporate waters. Using Enterprise Architects Decision Model and Notation (DMN) features, you can not only model your organization's decisions, but you can also run simulations from these models to predict

outcomes based on example data sets. The power of the language is that business people can readily understand and work with expressive but straightforward Decision Requirements diagrams that detail the decisions, including their inputs and the expected outputs. A modeler can document the rules in several ways, including easy to define decision tables. Once completed, these diagrams with accompanying input data examples, can be simulated to show the results of the decisions.

-



Publications and Documentation

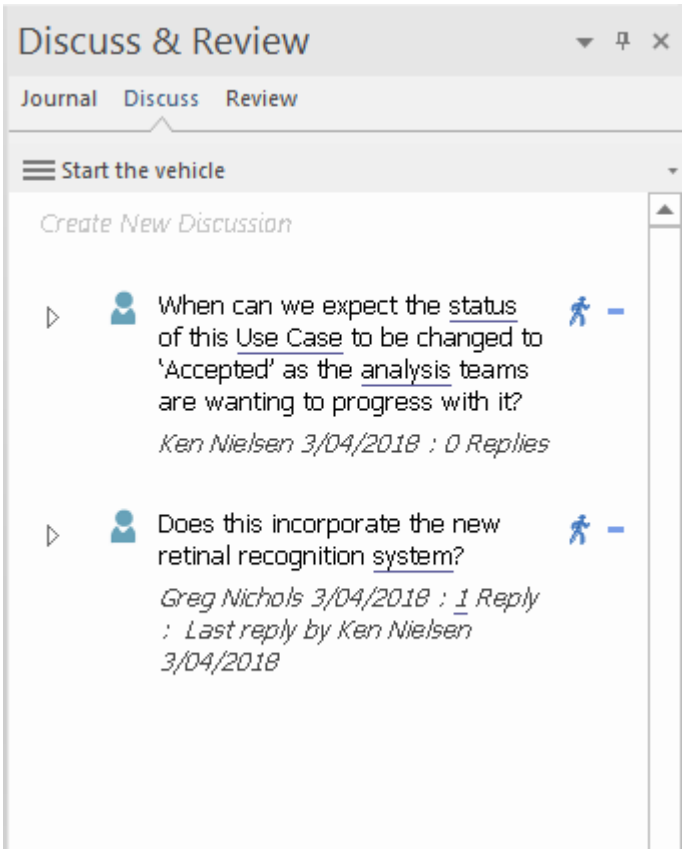
A systems engineer, manager, or other stakeholders can use the documentation features to automatically generate a wide range of documentation directly from the models. These can be document based such as PDF and Docx format or HTML-based. You can use flexible templates to completely tailor the generated documents, including company logos, tables of content, tables of element information, and diagrams. The success of a systems engineering project and, ultimately the entire engineering practice will depend on how well you communicate with stakeholders. Many stakeholders will be content to view systems models, including lists, diagrams, and matrices, directly in the repository. Still, others will want or require by contract, electronic, or printed documentation delivered to them. You can use the documentation generator to create high-quality corporate publications automatically from the repository. This includes a wide range of standard publications such as the Operation Concept document, Functional Architecture, Requirements Specification, and more. Modelers can also create Ad-hoc reports from tools such as the Glossary and the Search Window.

Collaboration and Teams

An engineering team is multidisciplinary and consists of strategists, managers, system engineers, software engineers, testers and others. The commercial pressures to release a product or provide a solution means that teams have to work more cleverly and cohesively to ensure engineering outcomes. Enterprise Architect has been built from the ground up as a collaborative platform, not just for engineers but for all disciplines. It facilitates individuals and teams working together and sharing information, models, designs, and solutions with a full range of tools from discussions, reviews, a team library, and chat to Version Control and Baselines.

Discussions and Chat

Central to the notion of collaboration is a modeler's ability to discuss and chat with colleagues or industry and standards specialists about a problem or solution. Enterprise Architect allows engineers, managers, and others to discuss elements, diagrams, and connectors. Any modeler can create a post to start a thread or conversation that other modelers can enter into by replying. The discussions are kept separately from element and diagram meta-information, allowing you to make rich and constructive comments without affecting documentation or reports generated from the models. The discussions and chat are two options available, discussions from the Discuss & Review window and chats from the Chat & Mail window.



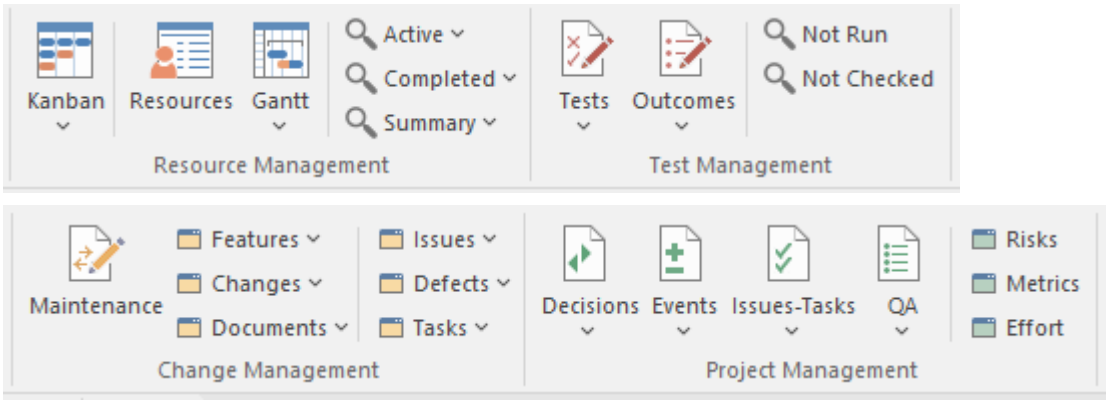
Chat is useful for quick and responsive communication with colleagues or experts that have been defined as part of a security defined group of users. Chats are not related to model elements in the way that discussions are but rather are global and when the Chat & Mail window is opened and a group is selected, the items are listed in date-time order. For more information see the Teams & Collaboration Help topic.

Project Management

Enterprise Architect has been built from the ground up with the Project Manager in mind. Systems repositories are valuable engineering assets and should be managed and maintained accordingly. Any team can assign themselves as a resource applied to an item performing particular roles, and you can conveniently view these in a built-in Gantt Chart. Risk can be modeled and managed in various locations, and engineers can determine project effort with built-in support for Metrics and Estimation. An Audit function allows changes to be tracked at a fine-grained level, and a Library facility and Element Reviews and Discussions enable users to work collaboratively on models.

The Construct ribbon provides a number of panels that contain project, resource and test management features including element maintenance items. The engineer can enter project management items at two levels:

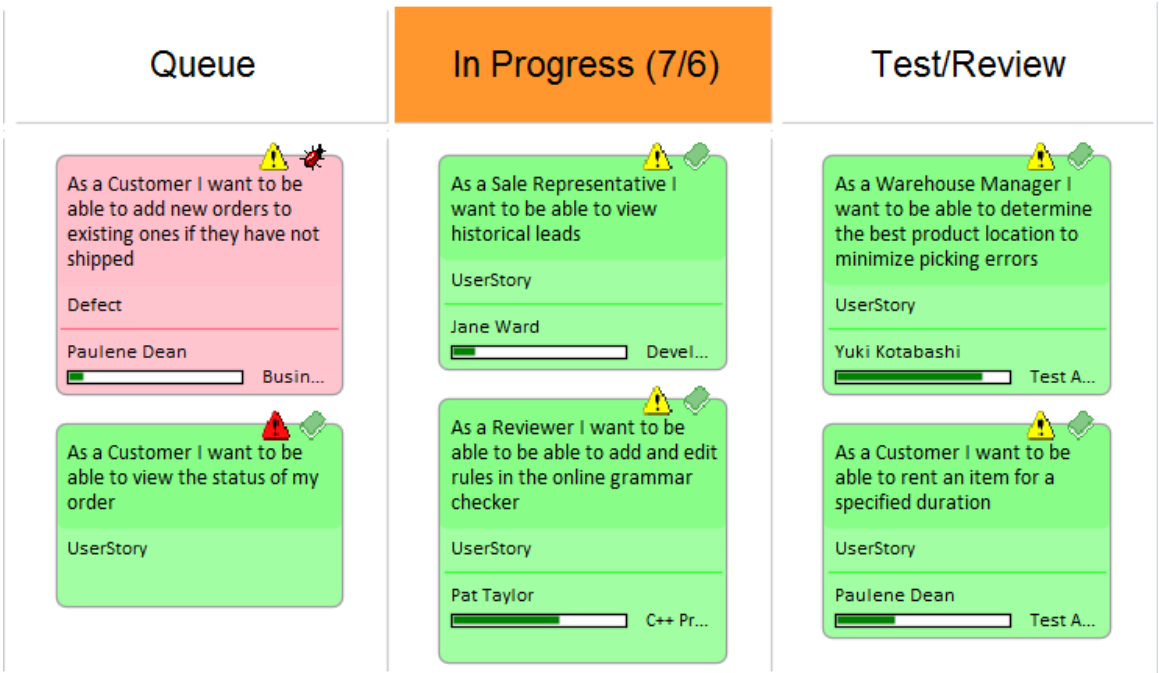
- Project level items - these apply at the level of the project.
- Element level items - these apply to specific elements.



The tool has a range of Project Management features for agile and iterative development as well as more traditional methods including those described here.

Kanban Diagrams

Engineers can utilize Kanban diagrams to manage part or all of an engineering project with more agility and flexibility. You can assign resources to various elements from Use Cases to Deployment Packages, and progress bars show each resource's percentage completion for each item.

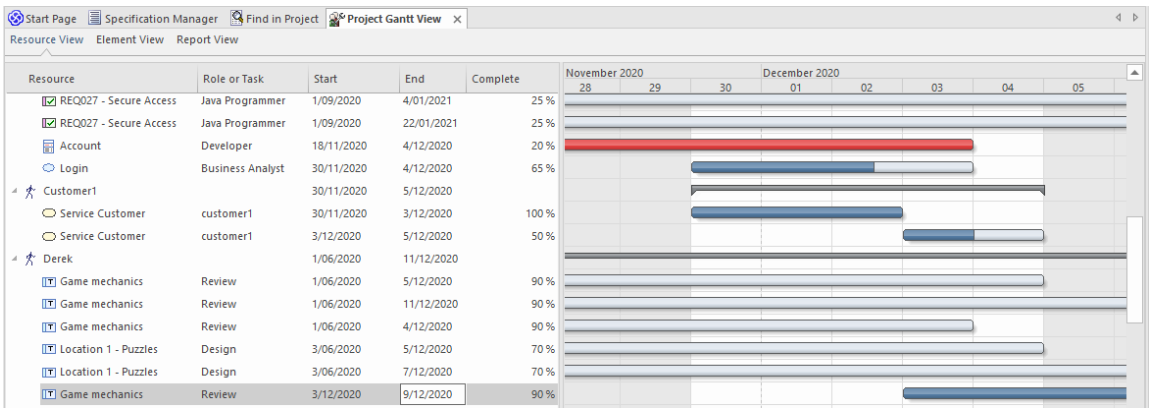


Resource Allocation and Gantt Charts

An engineering team comprises many different human resources, typically performing one or more roles in the engineering project. In developing a model, these resources create or maintain model structures and elements that describe or prescribe how the project will be run and the system built. As a Project Manager or engineer, you can assign resources to tasks on elements (including Packages) in the model, planning and monitoring their work within the timeframe you have allocated for that work to be completed. The allocation of resources can be carried out using the Project Gantt View, Kanban or Construct facilities.

The Gantt View is a tool that allows the engineer or project manager to visualize the elements in a project, Package, or diagram and the resources that have been allocated to them. There are several different Gantt charts available:

- The Project Gantt View used to view elements across the entire repository.
- Diagram Gantt View, which you use to display the allocation of resources to the elements in a given diagram.
- Package Gantt View, which you use to display the allocation of resources to the elements contained in a selected Package.



The tool will empower the traditional or agile Project Manager to ensure that a project's resources are allocated to repository content and help ensure high-value outcomes are achieved right from within the repository.

The Gantt View's primary use is to display the allocation of resources to elements in the repository and to manage the work breakdown structure. You can apply a wide range of views and filters to tailor the view or make it more relevant to a particular audience. Allocations can be made to any elements in the repository, from high-level Packages to an

individual element such as a Class, Activity, or Change. It is a powerful tool for Project Managers to visualize how a team can ultimately deliver high value and high priority outcomes. Modelers working on a project can view their own work and update their progress on assigned tasks.

While an engineer can make broad changes using the visual duration bars in the Gantt View, it is common practice to use the tool in conjunction with the Resource Allocation window where fine details can be entered and adjusted.

Change Management

Change is inevitable, and the best-run engineering projects predict that change will occur and plan for it. As an Engineer, Project Manager or Product Manager, you can use Enterprise Architect to register a range of maintenance items against any element in the project, including Requirements, Blocks, Activities, Use Cases, and Packages.

Maintenance items are defects, changes, issues, tasks, features, and documents that apply at the model element level. They are properties of individual model elements that the Engineer can use to record and capture problems, changes, issues, and tasks as they arise and document the solution and associated details.