VALU3S

Verification and Validation of Automated Systems' Safety and Security

V&V Modelling Language (VVML) Handbook

Document Type	Handbook
Primary Author(s)	Bob Hruska (LLSG), Wolfgang Herzner (AIT)
Document Date	31 May 2023
Document Version	2.9 Final
Dissemination Level	PU
Project Coordinator	Behrooz Sangchoolie, behrooz.sangchoolie@ri.se, RISE Research Institutes of Sweden
Project Homepage	www.valu3s.eu
JU Grant Agreement	876852

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

Executive Summary

This document is a guide to VVML, a domain-specific language (DSL) for modelling verification and validation (V&V) workflows. VVML has been developed in the EU ECSEL project VALU3S, which aimed to design, implement, and evaluate state-of-the-art V&V methods and tools in order to reduce the time and cost needed to verify and validate automated systems with respect to safety, cybersecurity and privacy requirements. Within the project, the generic V&V workflow design approach and modelling language VVML has been developed to easily visualize V&V-oriented workflows in industrial use cases and concrete tool chains and to facilitate the understanding, analysis, and improvement of these workflows.

The document introduces the VVML notation, its elements, and diagram types, and explains how V&V workflows can be modelled. It also presents, rules and guidelines for creating V&V workflows are introduced. VVML has been implemented as a plug-in to Enterprise Architect (EA), a UML modelling tool by Sparx Systems. It has been used to model all the V&V workflows of the 13 industrial use cases of the VALU3S project.

Contributors

Bob Hruska	LLSG	Wolfgang Herzner	AIT
Katia Di Blasio	INTECS	José Proença	ISEP
Robert Sicher	LLSG	Thomas Bauer	FRAUNHOFER IESE

Reviewers

Behrooz Sangchoolie	RISE	2023-03-21, 2023-05-31
Ugur Yayan	ESOGU	2023-03-22
Leire Etxeberria	MGEP	2023-03-22

Revision History

Version	Date	Author (Affiliation)	Comment
1.0	4.6.2021	Bob Hruska	Initial release
1.1	18.6.2021	Bob Hruska	Document styles changed, added change history chapter, modified Figure 8, enhanced description of a Fork / Join Node
2.0	24.2.2022	Wolfgang Herzner	Structure reworked, VVML-usage rules/guidelines added
2.1	7.3.2022	Bob Hruska	v2.0 reviewed
2.2	17.3.2022	Wolfgang Herzner	Rules/guidelines added, uploaded to SharePoint
2.3	20.5.2022	Wolfgang Herzner	Layout converted to VALU3S format
2.4	2.3.2023	Wolfgang Herzner, Bob Hruska, Katia Di Blasio, José Proença, Thomas Bauer	All chapters updated
2.5	23.3.2023	Behrooz Sangchoolie (RISE)	Collected review comments on the first draft of the handbook.
2.6	26.4.2023	Wolfgang Herzner, Bob Hruska, Katia Di Blasio, José Proença, Thomas Bauer	Final updates based on comments from first review

Version	Date	Author (Affiliation)	Comment
2.7	29.5.2023	Behrooz Sangchoolie (RISE)	Review of the second final draft of the report while making minor formatting changes.
2.9	31.5.2023	Behrooz Sangchoolie (RISE)	Final version of the report to be placed on the website and VALU3S web-based repository.

Consortium

RISE RESEARCH INSTITUTES OF SWEDEN AB	RISE	Sweden
STAM SRL	STAM	Italy
FONDAZIONE BRUNO KESSLER	FBK	Italy
KNOWLEDGE CENTRIC SOLUTIONS SL - THE REUSE COMPANY	TRC	Spain
UNIVERSITA DEGLI STUDI DELL'AQUILA	UNIVAQ	Italy
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO	ISEP	Portugal
UNIVERSITA DEGLI STUDI DI GENOVA	UNIGE	Italy
CAMEA, spol. s r.o.	CAMEA	Czech
IKERLAN S. COOP	IKER	Spain
R G B MEDICAL DEVICES SA	RGB	Spain
UNIVERSIDADE DE COIMBRA	COIMBRA	Portugal
VYSOKE UCENI TECHNICKE V BRNE - BRNO UNIVERSITY OF TECHNOLOGY	BUT	Czech
ROBOAUTO S.R.O.	ROBO	Czech
ESKISEHIR OSMANGAZI UNIVERSITESI	ESOGU	Turkey
KUNGLIGA TEKNISKA HOEGSKOLAN	KTH	Sweden
STATENS VAG- OCH TRANSPORTFORSKNINGSINSTITUT	VTI	Sweden
UNIVERSIDAD DE CASTILLA - LA MANCHA	UCLM	Spain
FRAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V.	FRAUNHOFER	Germany
SIEMENS AKTIENGESELLSCHAFT OESTERREICH	SIEMENS	Austria
RULEX INNOVATION LABS SRL	RULEX	Italy
NXP SEMICONDUCTORS GERMANY GMBH	NXP-DE	Germany
PUMACY TECHNOLOGIES AG	PUMACY	Germany
UNITED TECHNOLOGIES RESEARCH CENTRE IRELAND, LIMITED	UTRCI	Ireland
NATIONAL UNIVERSITY OF IRELAND MAYNOOTH	NUIM	Ireland
INOVASYON MUHENDISLIK TEKNOLOJI GELISTIRME DANISMANLIK SANAYI VE TICARET LIMITED SIRKETI	IMTGD	Turkey
ERGUNLER INSAAT PETROL URUNLERI OTOMOTIV TEKSTIL MADENCILIK SU URUNLER SANAYI VE TICARET LIMITED STI.	ERARGE	Turkey
OTOKAR OTOMOTIV VE SAVUNMA SANAYI AS - OTOKAR AS	OTOKAR	Turkey
TECHY BILISIM TEKNOLOJILERI DANISMANLIK SANAYI VE TICARET LIMITED SIRKETI - TECHY INFORMATION TECHNOLOGIESAND CONSULTANCY LIMITED COMPANY	TECHY	Turkey
ELECTROTECNICA ALAVESA SL	ALDAKIN	Spain
INTECS SOLUTIONS SPA	INTECS	Italy
LIEBERLIEBER SOFTWARE GMBH	LLSG	Austria
AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GMBH	AIT	Austria
E.S.T.E. SRL	ESTE	Italy
NXP SEMICONDUCTORS FRANCE SAS	NXP-FR	France
BOMBARDIER TRANSPORTATION SWEDEN AB	BT	Sweden
QRTECH AKTIEBOLAG	QRTECH	Sweden
CAF SIGNALLING S.L	CAF	Spain
MONDRAGON GOI ESKOLA POLITEKNIKOA JOSE MARIA ARIZMENDIARRIETA S COOP	MGEP	Spain
INFOTIV AB	INFOTIV	Sweden
BERGE CONSULTING AB	BERGE	Sweden
CARDIOID TECHNOLOGIES LDA	CARDIOID	Portugal

Table of Contents

List of Figures

List of Tables

Acronyms and Definitions

Action	either an → Activity or a → Nested Method.
Activity	An elementary processing step.
Artifact	A physical or logical item exchanged between → Activities and/or → Methods. Examples are requirements, test data or software.
Diagram	The diagram serves as a drawing surface in which model elements and connectors are visualized.
DSL	Domain-Specific Language
EA	Enterprise Architect, a UML modelling tool provided by Sparx Systems
MDG	Model-Driven Generation Technology. A Technology that allows users to extend Enterprise Architect modelling capabilities to specific domains and notations.
Metamodel	A metamodel is a "scheme" or template for creating models. It defines the individual model elements, their semantics and structure.
Method	Central concept in VVM. In the context of VVML, a method is an agglomeration of → Activities and/or nested Methods that serves a specific goal. Methods consume and produce → Artifacts.
Nested Method	a(nother) → Method that is inserted into the current workflow
Nesting Method	that → Method for which the current workflow is defined
SUT	System Under Test, the item that is subjected to V&V processes
UML	Unified Modelling Language
UML profile	Provides a generic extension mechanism for creating domain-specific languages.
V&V	Verification and Validation
VVML	Verification and Validation modelling language
Workflow	A graph describing what happens in a → Method or a use case to produce the expected output → Artifacts by consuming its input → Artifacts. It is depicted in a Workflow diagram.

Chapter 1 Introduction

This document is the handbook for modelling V&V workflows in EA 15 using the VVML framework, which has been created to be able to graphically describe workflows of V&V methods that will improve the time, effort, and cost of V&V processes.

1.1 Purpose

The purpose of this document is to introduce a recommendation based on best practices in workflow design and to describe how to apply it in practice. It contains the details of the key activities, instructions, conventions, notations, and expectations of VVML. This initiative is an enabler to present novel standards and workflows that can be used as references in industrial applications.

1.2 Document Structure

In Chapter 2, the modelling workflow and the modelling framework are introduced, including hints on how to start working with it, such as creating diagrams. Chapter 3 contains the rules and guidelines for creating consistent workflow diagrams. Chapter 4 and Chapter 5 give examples and present the corresponding parts of the DSL, e.g., metamodels, which is the basis of the modelling framework.

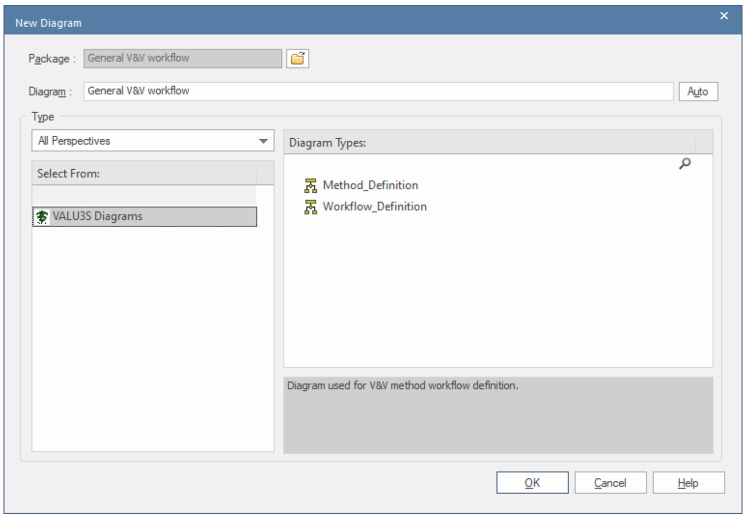
Chapter 2 VVML – a DSL for V&V Workflow Modelling

In modelling languages such as UML, it is possible to represent the same idea in many different ways. While the flexibility offered by the language is a positive aspect, it also creates problems in communicating ideas effectively. Not everyone is a UML expert or knows all the features that this modelling language offers. By creating a DSL that clearly specifies which diagrams and elements can be used in the creation of a V&V method definition or its workflow, everyone follows a common standardized language. The Modelling of V&V workflows falls into a specialized domain that requires a tailored modelling approach. To meet such requirements, there is the need to develop a UML profile for a V&V Modelling Language – shortly *VVML profile* – as a modelling framework that enables rapid modelling of V&V workflows.

2.1 Modelling Framework EA

The tool environment, in which VVML is implemented, is Enterprise Architect (EA) [1], a UML modelling tool from Sparx Systems. In EA, new modelling languages can be created with UML Profiles that can be used directly afterwards or can be packaged in an MDG-Technology for more comfortable use. MDG Technologies seamlessly plug into Enterprise Architect to provide additional toolboxes, diagrams, UML profiles, Shape Scripts, patterns, tagged values and other modelling resources. Such an MDG technology automatically generates a list of elements and relationships in the Diagram Toolbox.

Therefore, EA has been extended to provide a simple user-friendly interface for modelling V&V workflows with specially customized diagram types (see Figure 2.1), enabling workflow modelling with V&V methods, V&V work products, sequential control flows, quasi parallel control flow, fork and join, and flow of work products.


New Diagram

Package : General V&V workflow

Diagram : General V&V workflow

Type: All Perspectives

Select From: VALU3S Diagrams

Diagram Types: Method_Definition, Workflow_Definition

Diagram used for V&V method workflow definition.

Figure 2.1 VVML diagram types for modelling a V&V method definition and its workflow

2.2 Specification of VVML Workflows and Methods by Example

An example VVML method and its associated workflow diagrams can be found on the left and right side of Figure 2.2, respectively. We will use these diagrams to illustrate the core elements of these diagrams and to fix some terminology.

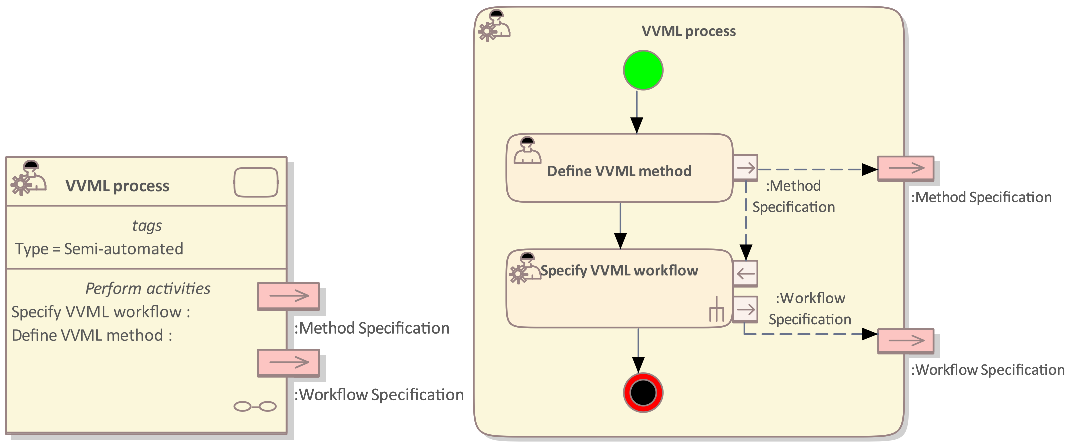


Figure 2.2 VVML Method Specification (left) and its corresponding Workflow Specification (right)

2.2.1 Method Specification

The left of Figure 2.2 presents a diagram specifying a VVML method named *"VVML process"*. It includes the following elements:

- **method type**, in this case *"semi-automated"*, capturing whether the method is automated, manual, or a mixture of both;
- **method artifacts**, named *"Method Specification"* and *"Workflow Specification"*, which are produced by the method; these can be input or output, although no input method artifacts exist in our example;
- **method activities**, listing the set of activities used inside this method, namely *"Define VVML method"* and *"Specify VVML workflow"* here, which are described in more detail in its corresponding VVML workflow diagram.

When modelling a VVML method the focus lies on the specification of input and output method artifacts of the method, and is meant to provide a quick overview over the method.

2.2.2 Workflow Specification

The method specified in the left of Figure 2.2 is defined using the workflow diagram on the right of the same figure (Intuitively, describing a workflow starts by defining a VVML method, followed by the specification of a VVML workflow, thus producing both a *"Method Specification"* artifact and a *"Workflow specification"* artifact.).

More specifically, this diagram includes the following elements:

- **start nodes**, depicted as green circles, represent the starting point of the workflow;
- **stop nodes**, depicted as red and black circles, represent the stopping point of the workflow;
- **activities**, depicted as yellow rectangles, named *"Define VVML method"* and *"Specify VVML workflow"* in our example, represent basic atomic steps, also labelled as (semi-) automated/manual. Activities with a fork symbol at the bottom right corner are instances of VVML methods, such as the *"Specify VVML workflow"* activity in Figure 2.2;
- **sequence-flow arrows**, depicted with solid lines, enforcing the order of execution of the activities;
- **artifact-flow arrows**, depicted with dashed lines, describing how artifacts are exchanged;
- **activity-artifacts**, denoted as small squares at the border of activities with an incoming or outgoing arrow, representing interfaces with required or produced artifacts, respectively.

Other elements exist, e.g., to support concurrent or alternative activities in a workflow, which will be presented later. The main purpose of the workflow specification is to orchestrate both the execution of the activities and the exchange of artifacts of a given process.

2.3 Creating a VVML Method

After you have started EA (with the VVML toolset installed), firstly open an existing package or alike. If none exist, consult the EA instructions to create a workspace or alike.

In the Toolbox menu, you should now have the option *"Package"*. Place it on the active drawing window (or on another package in the EA Browser window) and give it a proper name. Select *"Create Diagram"* for *"Initial Content"*.

You will be immediately prompted by the *"New Diagram"* window, as shown below in Figure 2.1. Under *"Select Forms"*, scroll until you find "VALU3S Diagrams". Click on it, and you will see two diagram types as shown in the figure.

Select *"Method_Definition"*. In the Browser window, an entry with the chosen name and the diagram symbol appears (see Figure 2.3).

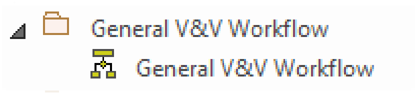

General V&V Workflow
General V&V Workflow

Figure 2.3 Result in Browser window

When you double-click on the diagram entry there, in the editing area of EA a new slot appears, and the Toolbox menu should show the elements depicted in Figure 2.4.

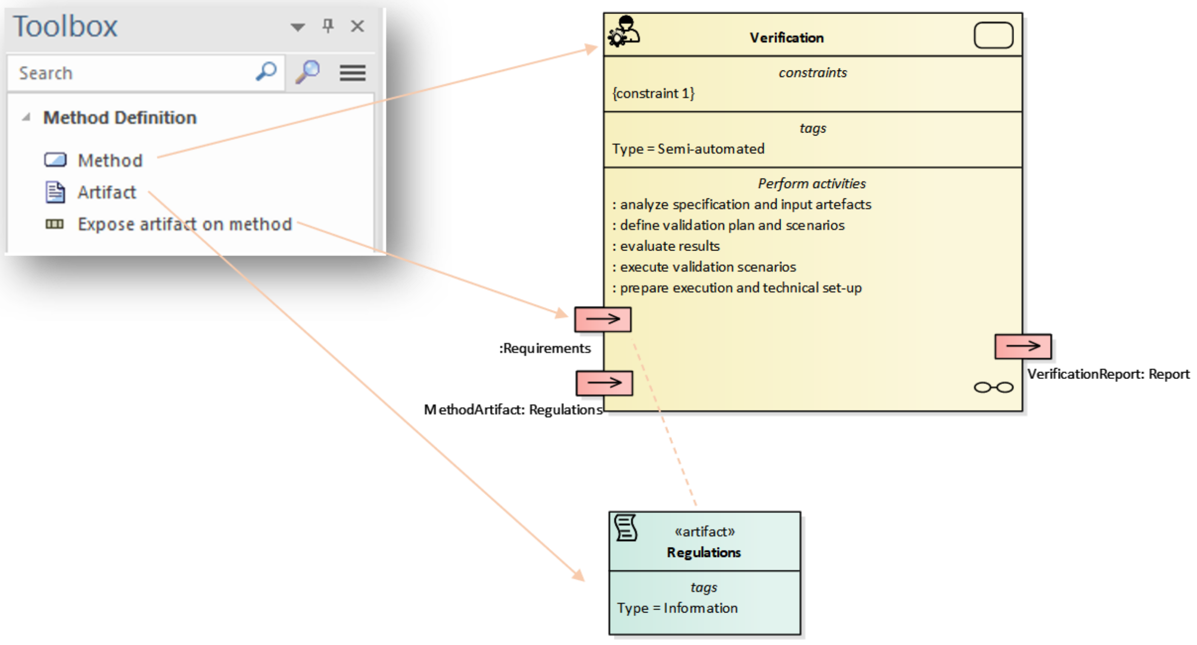

Toolbox
Search
Method Definition
Method
Artifact
Expose artifact on method
Verification
constraints
{constraint 1}
tags
Type = Semi-automated
Perform activities
: analyze specification and input artefacts
: define validation plan and scenarios
: evaluate results
: execute validation scenarios
: prepare execution and technical set-up
:Requirements
MethodArtifact: Regulations
VerificationReport: Report
«artifact»
Regulations
tags
Type = Information

Figure 2.4 Method definition diagram example with the dedicated toolbox and its toolbox items

Firstly, select the Method element and place it on the (empty) drawing area. You will be prompted to select its type among:

- *Manual*: choose this type if all tasks of it need to be done by humans
- *Automated*: choose this type if all tasks of it are executed automatically
- *Semi-automated*: otherwise

Note that, the type, as other properties, can be changed in the EA Properties window.

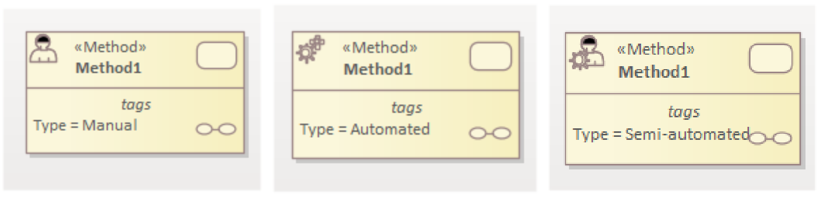


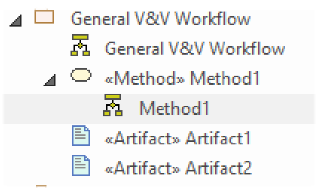
Figure 2.5 First Method image with selected type

As next step, you should define the artifacts the method uses. For that purpose, place the artefact element from the Toolbox on the drawing area; you will be prompted to select its type among:

- *Information*: a passive artefact carrying information,
- *Active Unit*: an active item, usually some soft- or hardware

Note again that the type, as other properties, can be changed in the EA Properties window.

You should do this last task for each of the artifact the method needs or produces. For each artifact, you will find an entry in the EA window Browser (Project tab), as that shown in Figure 2.6.

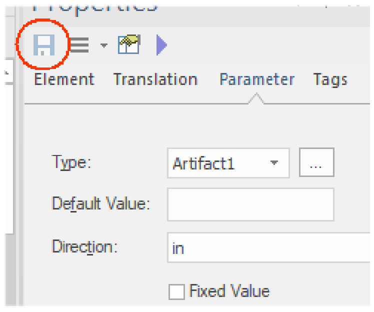


General V&V Workflow
General V&V Workflow
«Method» Method1
Method1
«Artifact» Artifact1
«Artifact» Artifact2

Figure 2.6 Browser window after generating a method and two artifacts

Finally, you should define which artefact is input to or output from the method. For each method artefact, this is done following the next steps:

1. Place *"Expose artifact on method"* on the (yellowish) method box. A pink rectangle with an arrow in it appears on its border (see Figure 2.7).
2. Select it.
3. In the Properties window of EA, select the tab "Parameter".
4. Click on the *"…"* right to *"Type [Integer]"*; a browser window opens.
5. Open the "Model" entry repeatedly until the Artifact just created appears (Figure 2.6) and select it.
6. In the same tab, select the *"Direction"*.
7. Important: click finally on the floppy disk symbol to save the settings (see Figure 2.7)!



Type: Artifact1
Default Value:
Direction: in
☐ Fixed Value

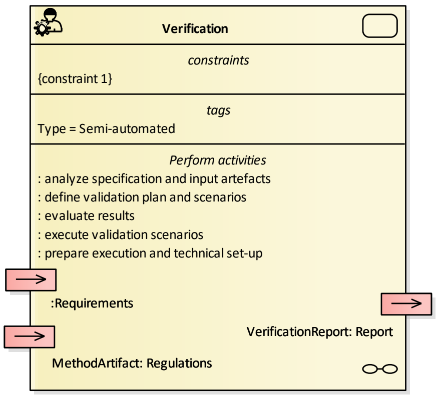
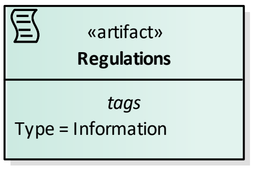
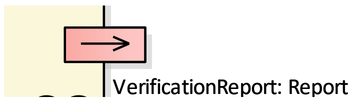
Figure 2.7 Properties window for defining Method artifact

2.4 Summary

In the following, the previous steps are summarized in a somewhat more systematic manner.

2.4.1 Method Definition Elements

Table 2.1 Method Definition Elements - Overview

Element	Notation	Description
Method	 Verification constraints {constraint 1} tags Type = Semi-automated Perform activities : analyze specification and input artefacts : define validation plan and scenarios : evaluate results : execute validation scenarios : prepare execution and technical set-up :Requirements VerificationReport: Report MethodArtifact: Regulations	Represents a high-level definition of a «Method». It specifies the activities contained in its workflow process, the «Artifact» it consumes and produces, and other information such as its processing type. The icon in the top left corner indicates the method's processing type: • Manual • Automated • Semi-automated
Artifact		Represents an information object, e.g., a document, or a functional (active) unit, e.g., a SUT, either produced (output) or consumed/used (input), by a «Method» / «Activity» (depending on the level of abstraction). The icon in the top left corner indicates the artifact type: • Information • Active Unit
Expose Artifact on Method **(MethodArtifact)**		A «MethodArtifact» is a structural element representing an exposed «Artifact» with a set direction in/out. This indicates which artifacts are provided (output) or required (input) by a method.

2.4.2 Steps for Method Definition

1. When defining a method, the user should follow the steps below:In an existing EA-package, create a VVML Method definition diagram with name <method> (from *"VALU3S Diagrams"*, under *"New Diagram" / "Select Forms"*).
2. Create a VVML «Method»: select the method type and name after the diagram name.
3. Use the notes field of the VVML «Method» element for a brief textual description of the method.
4. Create a VVML «artifact»: select the artifact type, give it a name and use the notes field of the artifact for a brief textual description.
5. Create a structural element VVML «MethodArtifact» of the VVML «Method» created in step 2.
6. Type the VVML «MethodArtifact» by the VVML «artifact» created in step 4.
7. Define a direction of the VVML «MethodArtifact».
8. Repeat steps 4-7 until all the «artifact» elements of the method are identified.

Figure 2.8 depicts an overview of the steps needed to define a workflow specification using VVML. Note that valid workflow specifications must also include artifacts and all internal activities must produce at least one artifact. This will be further explained later in this document. However, most of the following examples will omit artifact information to simplify the presentation.

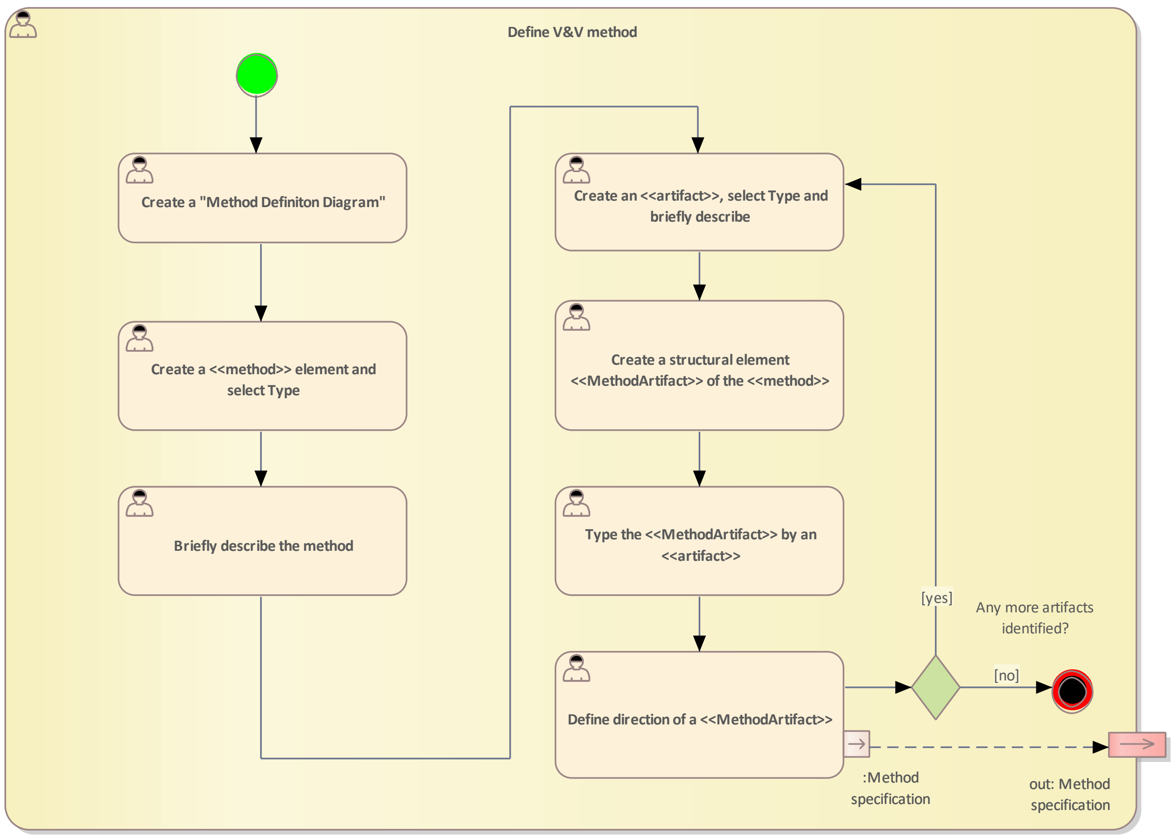
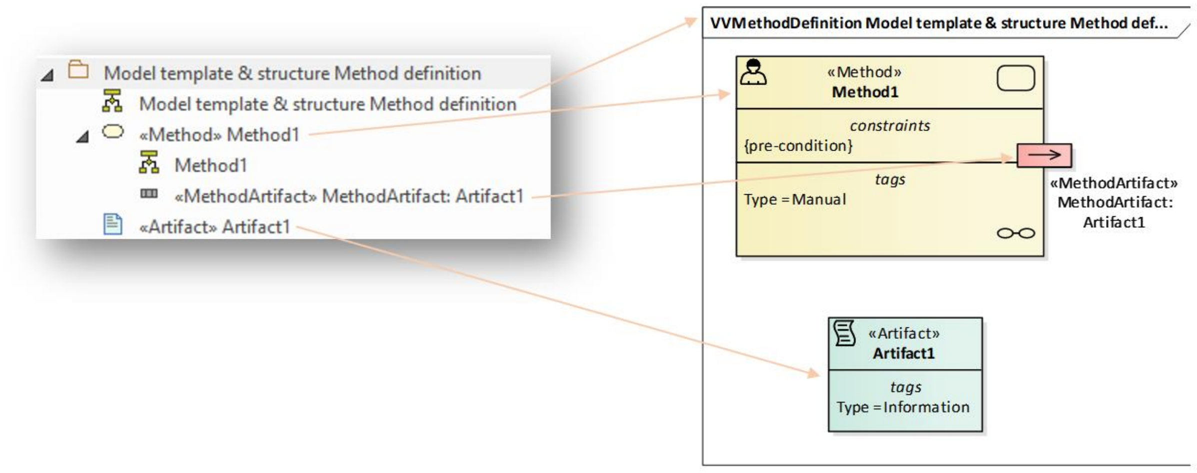

Define V&V method
Create a "Method Definiton Diagram"
Create a <<method>> element and select Type
Briefly describe the method
Create an <<artifact>>, select Type and briefly describe
Create a structural element <<MethodArtifact>> of the <<method>>
Type the <<MethodArtifact>> by an <<artifact>>
Define direction of a <<MethodArtifact>>
[yes]
Any more artifacts identified?
[no]
:Method specification
out: Method specification

Figure 2.8 VVML Workflow definition diagram describing the steps needed for a creation of a «Method» element

Figure 2.9 shows where the elements needed for a Method definition are located in the model structure (on the left) and graphically visualized in a diagram (on the right). Note that making the element «artifact» visible on a diagram is optional.



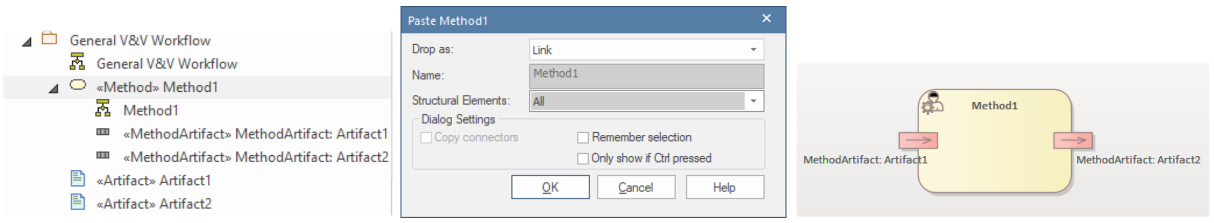
Model template & structure Method definition
Model template & structure Method definition
«Method» Method1
Method1
«MethodArtifact» MethodArtifact: Artifact1
«Artifact» Artifact1
VVMethodDefinition Model template & structure Method def...
«Method»
Method1
constraints
{pre-condition}
tags
Type = Manual
«MethodArtifact»
MethodArtifact:
Artifact1
«Artifact»
Artifact1
tags
Type = Information

Figure 2.9 Visual representation of the Method definition diagram and its elements structure in the model package structure

2.5 Specifying a VVML Method Workflow

For starting the specification of the method workflow, you should double-click on the yellowish method box (in the editing area of EA). A new empty tab will be opened in the editing area of EA, with the method name as tab name.

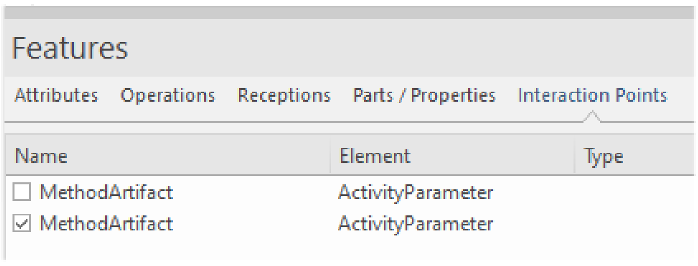
Now, drag the Method entry (*"«Method» Method1"* in Figure 2.10) from the EA Browser window onto the empty editing area (Figure 2.10, left). You will be prompted with the *"Paste <<Method name>>"* window, where you must set *"Drop as" to "Link"* and *"Structural Elements"* to *"All"* (Figure 2.10, centre). The bounding box where to draw the workflow will appear in the drawing/editing area (Figure 2.10, right).



Drop as: Link
Name: Method1
Structural Elements: All
Copy connectors
Remember selection
Only show if Ctrl pressed

Figure 2.10 Creating a method's workflow frame

Note: if not all MethodArtifacts are shown, do the following: select tab Interaction Points in the EA window Features. Here, select all list entries not having the selection box filled. In Figure 2.11, this is the case for the first entry.



Features

Attributes | Operations | Receptions | Parts / Properties | Interaction Points

	Name	Element	Type
☐	MethodArtifact	ActivityParameter	
☑	MethodArtifact	ActivityParameter	

Figure 2.11 Example of unselected MethodArtifact in EA Features window

Now, the workflow can be created. For that purpose, the EA Toolbox windows shows different items as for method definition, as shown in Figure 2.12.

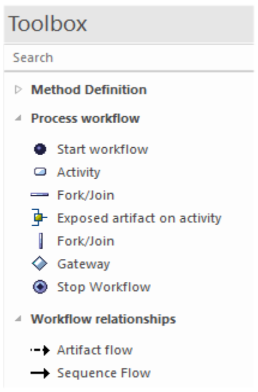


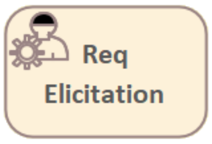
Figure 2.12 Items for method workflow definition

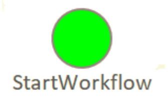
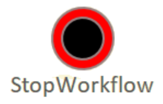
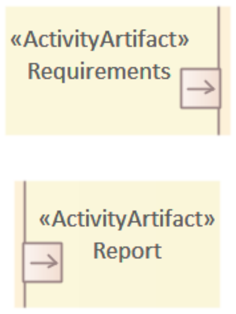
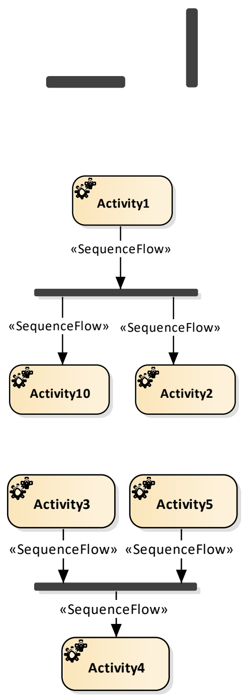
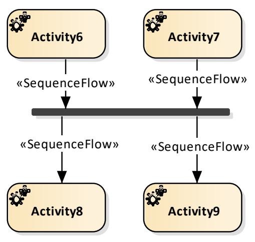
In the following, the meaning and handling of these elements is described, with the help of concrete examples. For their correct use, i.e., for the creation of correct workflows, see Chapter 3. For further hints, see Chapter 5.

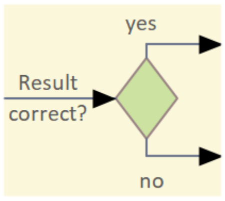
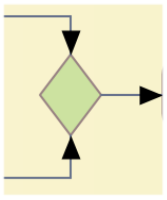
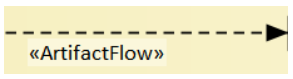
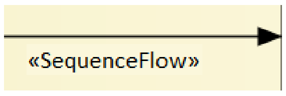
2.6 Workflow Specification Elements

The elements for workflow specification are described in Table 2.2 below.

Table 2.2 Workflow Specification Elements - Overview

Element	Notation	Description
Activity		Represents a basic (atomic) step within a method workflow and is always owned by a «Method» element. Just like «Method», the icon in the top left corner indicates the «Activity» type: • Manual • Automated

Element	Notation	Description
		• Semi-automated
StartWorkflow		A node that initiates a workflow.
StopWorkflow		Indicates the termination of a Method workflow: upon reaching the Stop Workflow, all execution in the Method workflow diagram is aborted.
Exposed Artifact on Activity **(ActivityArtifact)**	 «ActivityArtifact» Requirements «ActivityArtifact» Report	An «ActivityArtifact» is a structural element representing an exposed «Artifact» on an «Activity» element. The direction flow is indicated as for «MethodArtifacts». To create an «ActivityArtifact», place the symbol on the respective «Activity», move it to the requested position, and select the direction in the Kind field in the Element tab of the EA Properties window.
Fork/Join	 Activity1 «SequenceFlow» «SequenceFlow» «SequenceFlow» Activity10 Activity2 Activity3 Activity5 «SequenceFlow» «SequenceFlow» «SequenceFlow» Activity4	The VVML Fork Node is a control node that has one incoming edge and multiple outgoing edges and is used to split incoming flow into multiple concurrent sequence flows. Join Nodes have multiple incoming edges and one outgoing edge and are used to merge concurrent sequence flows. They can also be used to synchronize concurrent sequence flows at a certain point in the workflow, as the figure below shows.  Activity6 Activity7 «SequenceFlow» «SequenceFlow» «SequenceFlow» «SequenceFlow» Activity8 Activity9

Element	Notation	Description
Gateway	 	The VVML Gateway serves for creating or joining alternative control flows. For creating alternative flows, it is associated with a condition label. The process flow continues along the outgoing branch with the corresponding guard. The condition should always end with the character '?', and each outgoing edge has a guard indicating the alternative. Enter the label name in the field "Name" in the EA Properties window (tab Element). The lower diagram depicts the usage for joining alternative control flows. The outgoing branch is passed whenever the flow arrives via one of the incoming branches. In this case, leave the "Name" field empty.
ArtifactFlow		Artifact Flow connects two VVML Activities using the «ActivityArtifact» elements, indicating a specific VVML Artifact passing through it.
SequenceFlow		The Sequence Flow is a connector connecting two VVML Actions in a Method workflow diagram, modelling an active transition. Sequence Flow connectors bridge the flow between VVML Actions, by directing the flow to the target diagram element once the source node action is completed.

2.7 How to model a Method Workflow

When defining the method workflow, the user should follow the steps below:

1. Change the name of the existing VVML Method workflow diagram owned by the «Method» to match the <method> name. Please note that the Composite diagram was automatically created while creating the element «Method».
2. Create such a diagram boundary by dragging the VVML «Method» itself onto the child Composite diagram from the Browser and dropping it as a Link.
3. Use Features window to make all «MethodArtifact» (structural elements) of the «Method» visible by checking the listed elements in the window.

4. Model workflow applying the principles of a UML Activity diagram, but adding VVML stereotyped elements and relationships specified in the previous paragraph, and following the rules given in Chapter 3.
5. For each activity in the workflow diagram, use the notes field for a brief textual description of the activity.
6. Repeat step 4 until all the «Activity» elements of the workflow are identified.

Figure 2.13 depicts an overview of the steps described above needed for a workflow specification using VVML.

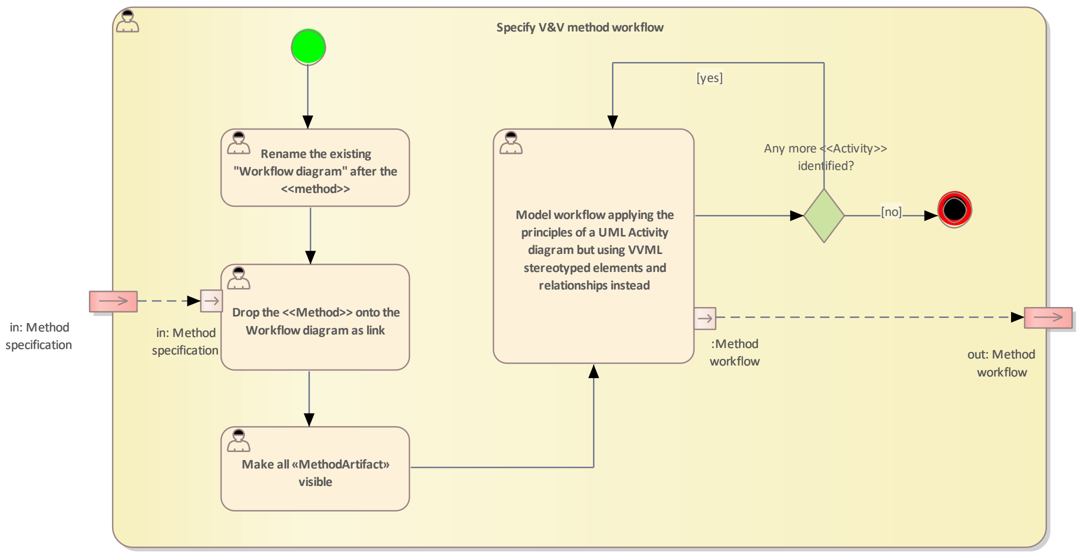

Specify V&V method workflow
Rename the existing "Workflow diagram" after the <<method>>
Drop the <<Method>> onto the Workflow diagram as link
Make all «MethodArtifact» visible
Model workflow applying the principles of a UML Activity diagram but using VVML stereotyped elements and relationships instead
Any more <<Activity>> identified?
[yes]
[no]
in: Method specification
in: Method specification
:Method workflow
out: Method workflow

Figure 2.13 VVML Workflow definition diagram describing the steps needed for a creation of a method workflow

Figure 2.14 shows where the elements needed for a Method workflow definition are located in the model structure (on the left) and their graphical visualization in a diagram (on the right).

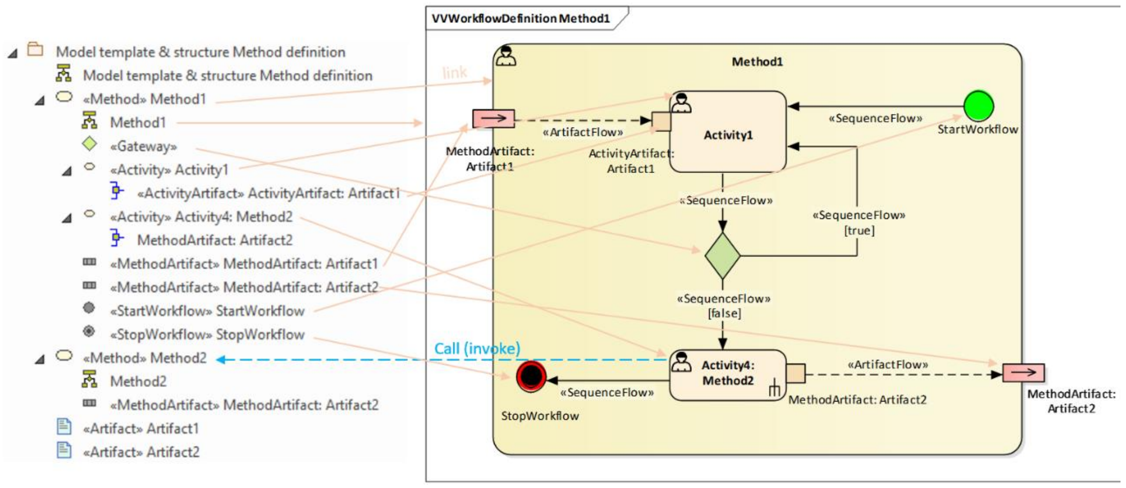
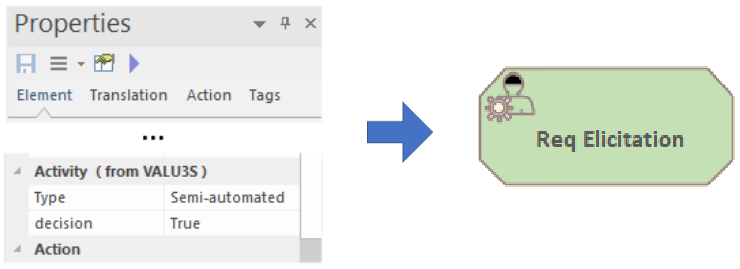

Model template & structure Method definition
Model template & structure Method definition
«Method» Method1
Method1
«Gateway»
«Activity» Activity1
«ActivityArtifact» ActivityArtifact: Artifact1
«Activity» Activity4: Method2
MethodArtifact: Artifact2
«MethodArtifact» MethodArtifact: Artifact1
«MethodArtifact» MethodArtifact: Artifact2
«StartWorkflow» StartWorkflow
«StopWorkflow» StopWorkflow
«Method» Method2
Method2
«MethodArtifact» MethodArtifact: Artifact2
«Artifact» Artifact1
«Artifact» Artifact2
VVWorkflowDefinition Method1
link
Method1
MethodArtifact: Artifact1
«ArtifactFlow»
ActivityArtifact: Artifact1
Activity1
«SequenceFlow»
StartWorkflow
«SequenceFlow»
«SequenceFlow» [true]
«SequenceFlow» [false]
Call (invoke)
Activity4: Method2
«ArtifactFlow»
«SequenceFlow»
StopWorkflow
MethodArtifact: Artifact2
MethodArtifact: Artifact2

Figure 2.14 Visual representation of the Workflow specification diagram and its elements structure in the model package structure

2.8 Decision Activities

Sometimes, an Activity mainly serves for preparing the decision following directly afterwards. In this case, that activity could be turned into a *"Decision Activity"*. This is done by simply setting its property *"decision"* in the Properties window (Element tab) to *"true"*, which turns its appearance into that shown Figure 2.15.


Properties

Element | Translation | Action | Tags

...

Activity (from VALU3S)	
Type	Semi-automated
decision	True
Action	

Figure 2.15 Turning an Activity into a Decision Activity

2.9 Using other Methods as Activities ("Nesting Methods")

Besides elementary Activities, it is also possible to use another VVML Method (that is described by its own workflow definition diagram) as an activity in a workflow. In other words, the current method *calls* this other method. In the workflow of the calling method, the called method is represented as *VVML CallBehavior Activity*, denoted by the symbol ⋔ in the lower right of the Activity symbol. Figure 2.16 illustrates a CallBehaviour on Activity4.

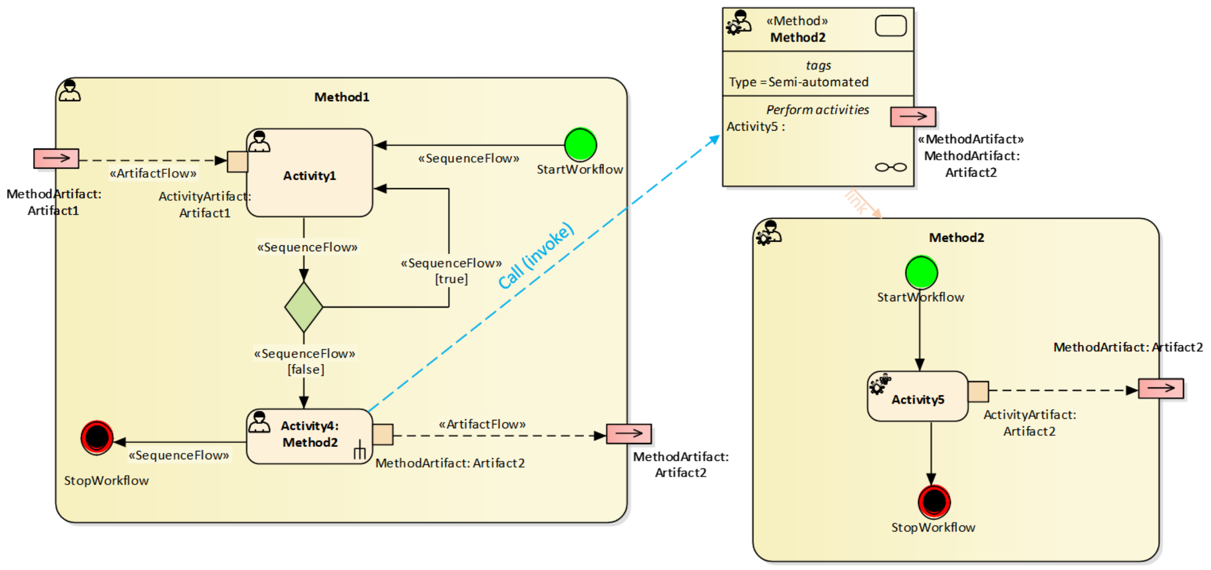


Figure 2.16 Illustration of the concept of calling a VVML method in a workflow

2.9.1 How to invoke an existing «Method»

1. From the Browser drag the VVML «Method» itself onto the Workflow definition (child Composite diagram) and drop it as an Invocation (Action). Do not change any other field in the *"Paste"* dialogue.
2. Use Features window to make all MethodArtifacts (column Stereotype) visible by checking the listed elements in the window. This happens analogously to the case when not all MethodArtifacts are shown after creating the workflow diagram (see Figure 2.11).

Chapter 3 Rules and Guidelines

This chapter covers rules and guidelines for modelling V&V workflows. Two types of recommendations are distinguished: **rules** shall, and **guidelines** should be considered.

Further note that the term *Sequence flow* is used for both the corresponding workflow diagram element and the transition of execution from the Action *A* at the root to the Action *B* at the end of such an arrow, as soon *A* terminates. For the latter semantics, lower case is used.

Correct workflows: most rules presented in this section prevent the definition of many *incorrect* workflows. Informally, a workflow is **correct** if:

- it never blocks before reaching the stopping node;
- it never reaches the stopping node while some activity is still running;
- it can always reach the stopping node;
- it never re-enters a running activity; and
- is able to start all of its activities.

Rules from this section are syntactic and often informal. When the rules over sequence flows are violated, the workflow is incorrect, although it is possible to specify incorrect workflows that obey these rules.

Please note that in the diagrams shown in this chapter <<ActivityArtifacts>> are usually dismissed if not relevant for the given aspect. Further, they do not show direction arrows.

3.1 General Rules

1) Distinguish clearly between Sequence Flow(s) and Artifact Flow(s).

The Sequence Flow defines the execution order of actions (Activities and nested Methods) within a method workflow, whereas the Artefact Flow defines the exchange of artefacts between actions or the exchange of artifacts, i.e., information, of a method with its environment. Accordingly, the graphical notation between Sequence Flows and Artifact Flows is clearly to be distinguished. Figure 3.1 illustrates the different elements.

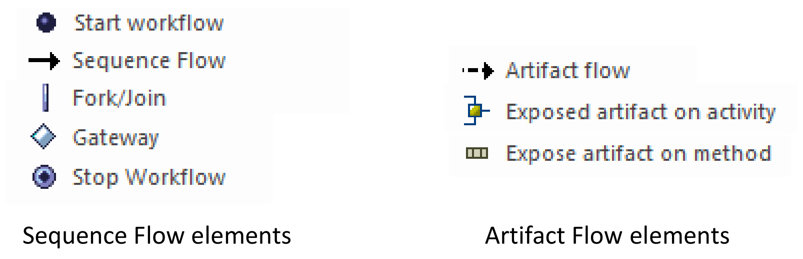


Figure 3.1 Elements of sequence flow and artefact flow

2) Distinguish between parallel and alternative Control Flows.

Parallel Control Flows are carried out simultaneously. They are always started and terminated with Fork and Join symbols, respectively. See rules 6) and 7) for details.

From alternative Control Flows, exactly one is executed. They are always started with either the Gateway symbol – see rule 8) – or using a *"decision activity"* – see rule 20). Note that, according to the fact that an Action is executed whenever a Control Flow arrives it (see rule 17), for joining alternative Control Flows the Gateway symbol is not required (see rule 21).

3.2 Rules for Sequence Flows

3) Control flow always starts at precisely one "Start Workflow" and terminates at precisely one "Stop Workflow".

If, due to a fork, several *"final"* actions exist, they have an outgoing sequence flow to a Join bar, from which the sequence flow leads to the Stop Workflow symbol. Figure 3.2 illustrates the use of start and stop nodes for workflows.

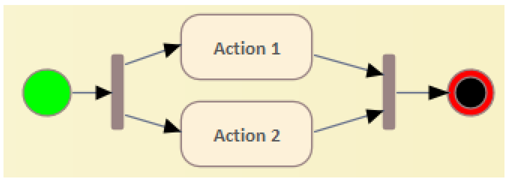
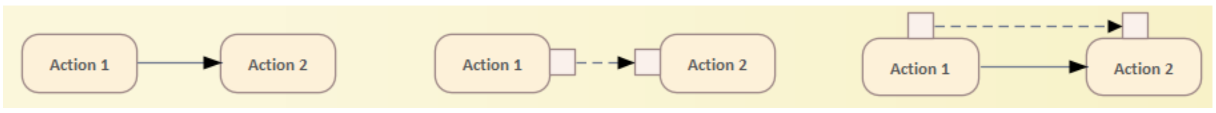


Figure 3.2 Start and stop nodes in sequence flows

4) Use always Sequence Flow to connect activities within a workflow.

As just stated, the Sequence Flow defines the execution order of actions (Activities and nested Methods) within a method workflow, not the arrival of input artifacts. In this way, this Sequence flow defines the synchronization between actions. Therefore, each action within a workflow has at least one incoming and one outgoing Sequence Flow. Figure 3.3 illustrates the different elements.



Correct Incorrect (without Sequence Flow) Correct
(At left image, Artifact flows dismissed by purpose; see also Rules for Artifact flows 3.3)

Figure 3.3 Using sequence flows to connect activities

5) Always one Sequence Flow leaves an Action.

In principle, precisely one Sequence Flow must emanate from an Action. Use *"Fork"* to start parallel control flows (see rule 6), and *"Gateway"* to start alternate control flows (see rule 8).

There is one exception, however, when using the *"Decision Activity"* (see rule 20).

6) Use "Fork" to start parallel Sequence Flows.

If, after an action *A*, several actions *B1 … Bn* should start in parallel, use the Fork symbol rather than letting several Sequence flows emanate from A (to the *Bi*). Figure 3.4 illustrates correct and incorrect examples for starting parallel sequence flows.

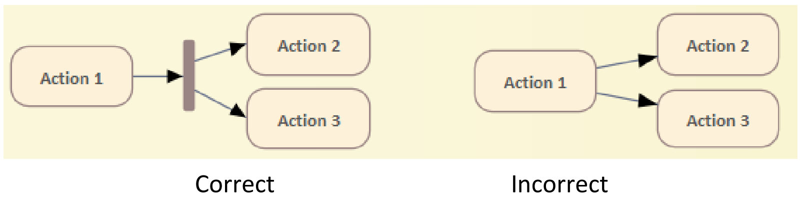


Figure 3.4 Using fork to start parallel sequence flows

7) Use "Join" to merge parallel Sequence Flows.

If, after termination of several actions *A1, ..., Ak,* an action *B* shall start, use the Fork symbol rather than letting several Sequence flows terminate in *B*. Figure 3.5 illustrates correct and incorrect examples for merging parallel sequence flows.

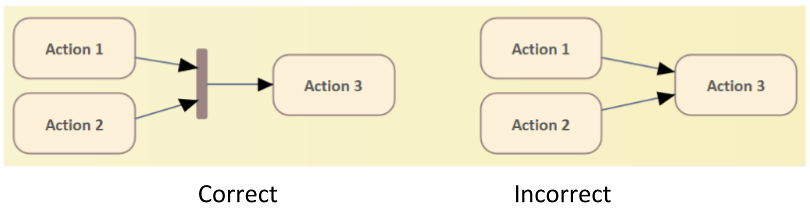


Figure 3.5 Using Join to merge parallel sequence flows

Action 3 will not start before both Action 1 and 2 finished. See rule 17) for the case where an action *B* shall start whenever one of several actions *A1, ..., Ak* terminates.

8) Use "Gateway" to start alternative Sequence Flows.

"Gateway" not only can be used for yes/no-decisions, but, in principle, also for n-way decisions. Therefore, if after an action *A* exactly one out of several actions *B1, …, Bn* should start, use the Gateway symbol (rather than letting several Sequence flows emanate from *A* (to the *Bi*)). In this case, you must also label each outgoing Sequence Flow properly. Figure 3.6 illustrates correct and incorrect examples for implementing alternative sequence flows.

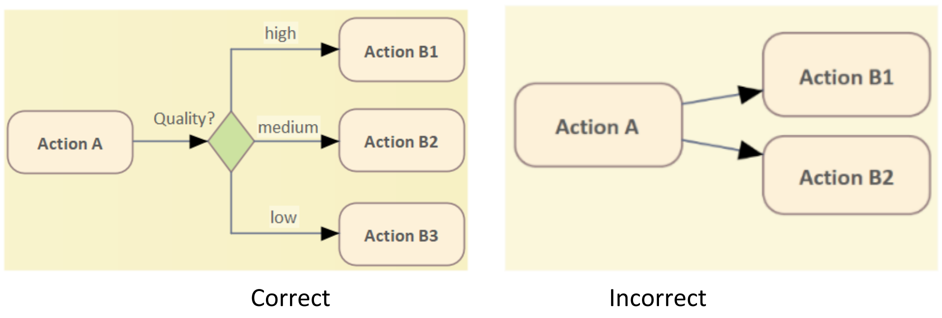


Figure 3.6 Implement alternative sequence flows

Please note that artifact flows are again dismissed to keep the examples simple.

For joining alternative Control Flows, see rules 2) and 21).

9) Use "Gateway" for Iterations.

If an action shall repeatedly be executed until a termination condition holds, use Gateways to this. Figure 3.7 illustrates correct and incorrect examples for implementing iterations in sequence flows.

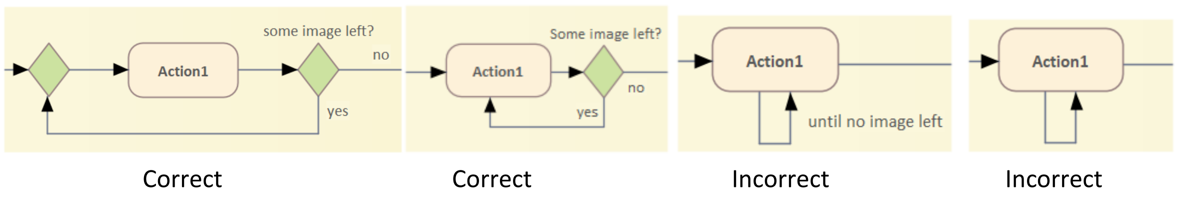


Figure 3.7 Implement iterations in sequence flows

Please note that artifact flows are again dismissed to keep the examples simple.

Please further note that the *"Decision Activity"* (rule 20) cannot be used in this case.

3.3 Rules for Artifact Flows

10) Artifact flows occur always between 'Pins'.

For each artifact of an action, an own pin is attached to it, and Artifact flows connect such pins. Pins are named *"Exposed artifact on activity"* in the VVML toolbox (see also rule 1).

Note that EA/VVML automatically generate pins if you try to connect two Activities directly with an Artifact flow.

11) Each Pin must have an Artifact Flow.

Activities must not have pins without any incoming or outgoing Artifact Flow connected. Figure 3.8 shows an example of an activity with an unconnected pin.

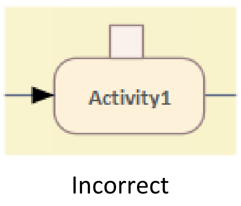


Figure 3.8 Example activity with unconnected pin

See rule 19) for pins of nested methods.

12) Each activity (and method) has at least one output artefact.

Actions work on artefacts and shall achieve something. Therefore, an activity without output is considered as invalid. Figure 3.9 shows example of activities with or without output artifacts.

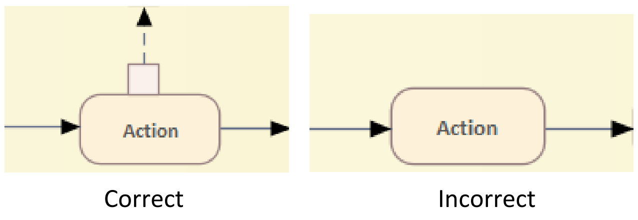


Figure 3.9 Example activities with and without output artifacts

Exception: Decision activities (see rule 20) have the decision as output; hence they do not need to have an explicit output artifact. This also implies that if an Activity preceding a Gateway symbol in Sequence flow does not have an Output artifact, rule 20) must be applied.

13) Copying of Artifacts.

If an output artefact is used by several activities, draw the Artifact flows directly from the respective output pin (of the producing action) to the input pins of the consuming actions, rather than using the Fork symbol (Fork/Join is only for Sequence flows.) Figure 3.10 shows how multiple copies of artifacts are used in workflows.

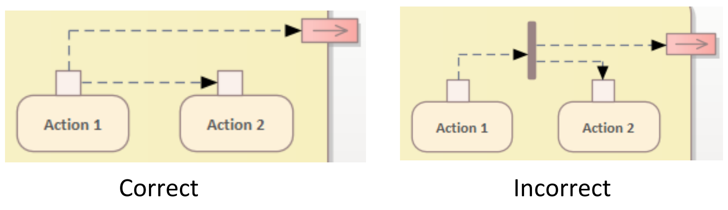


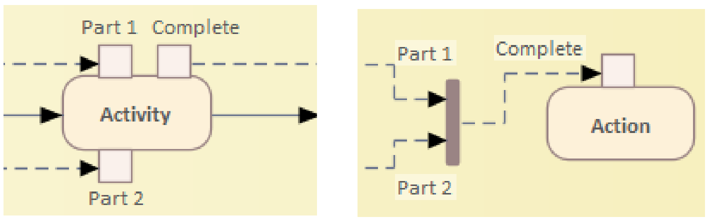
Figure 3.10 Using multiple copies of artifacts in workflows

Notes:

- Sequence flows have been left out only for simplicity.
- In these examples, the provision of an output artifact both to an internal Activity and as output Method Artifact is shown, but of course the same applies if several internal actions shall receive that artifact.

14) Merging of Artifacts.

Sometimes, it is necessary to combine artifacts into a new one. Although this may be as simple as concatenating several artifacts (parts) into one representing, e.g., a merged complete report, this always requires an own Activity. Hence, using e.g., Join bar for indicating such a concatenation is not allowed. Figure 3.11 shows how multiple artifacts can be merged into a single artifact.



Correct Incorrect

Figure 3.11 Merging artifacts

15) Mandatory use of Method Input Artifacts, and provision of Output Artifacts.

Each input artifact of a method must be the source of at least one Artifact flow to an (action) pin. Likewise, each output artifact of a method must be the target of precisely one Artifact flow from an (action) pin.

3.4 Rules for Workflow Execution

16) The execution of an activity or a nested method is considered as "atomic".

When the sequence flow arrives at an action, it starts, consuming the latest version of input artefacts (see also next rules). During its execution, it does not exchange artefacts with the environment. As soon as its execution stops, the latest versions of its output artifacts is delivered to the environment. Note that this does not mean that the activity execution takes no time (but from synchronization point of view it does).

<u>Note</u> that this will make the FAQ entry *"Synchronizing Artefact Flow with Control Flow"* in the handbook obsolete.

17) An Action is executed whenever any of the previous Actions is terminated.

If an action is part of a control flow loop or has several direct predecessors that are not synchronized with the *"Join"* bar (see rule 6), it is executed whenever any of the sequence flows arrives at it (see also rule 21).

If a Sequence flow from a *"Join"* bar arrives at an action, this action is executed as soon as the last Sequence flow ending at that Join bar is terminated.

18) Usage of Input Artifacts.

- <u>Always, the latest version of an artifact is used</u>.
  If, perhaps due to an iteration in a loop of previous activities, an action receives several versions of the same artefact, the last version is used. This applies also in case an action receives the same artefact from several activities (see also rule 18).
- <u>Reuse of latest artifact version</u>.
  If an action is executed several times and does not receive a new version of an artifact between two executions, the version used in the previous execution is reused.

- Defaults.
  If an action does not receive any version of a certain input artifact before its first execution, it uses a default version for that artifact.

19) Treating unused Pins of nested Methods.

In principle, the input artifacts of a nested method need to be fed with corresponding artefacts, either produced by some action of the nesting method, or by propagating some of their input artefacts. If this is not possible, it must be assured that the nesting method uses appropriate defaults.

If some output artefact of a nested method is not used by the nesting method, the corresponding output pin of the nested method should be hidden.

3.5 Guidelines for Graphical Simplifications

20) Use "Gateway" or Activity with Property *decision* = *True* to start alternative Sequence Flows.

If, after an action *A*, one of actions *B1*, ..., *Bk* shall be executed, depending on a certain decision, usually the Gateway symbol is used. However, if that decision is a direct consequence of Activity *A*, you can set the property *"decision"* of *A* to True, and directly connect via Sequence flows with *B1*, ..., *Bk*. Figure 3.12 shows examples of how activities can be combined with decisions.

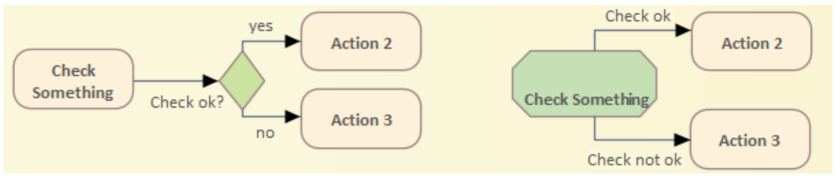


Figure 3.12 Combining activities with decisions

Please note that all outgoing Sequence flows are appropriately labelled.

21) Merging Gateway can be left out.

If an activity *B* shall start as soon as at least one some activities *A1*, …, *An* are executed, you could use the Gateway symbol receiving the sequence flows from the alternative activities *Ai* and propagating it to *B*. However, you could lead the sequence flows directly to *B* because, whenever the sequence flow arrives at an activity, it is executed. Figure 3.13 shows examples of how sequence flows can be merged to a target activity.

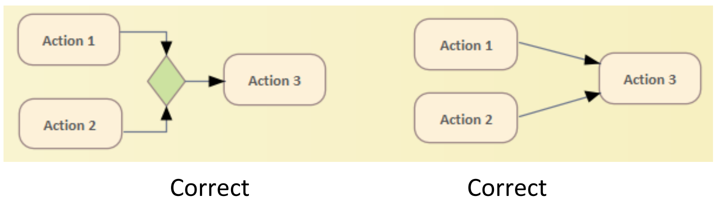


Figure 3.13 Merging sequence flows

22) Same Input Artifacts from several Sources.

If some actions can receive the very same input Artifact from several sources, the respective Artifact flows can end in the same pin (e.g., test cases provided by different test case generation methods). According to rule 18), at each execution of that action, the latest artifact version is used. Figure 3.14 shows an example of how an activity can receive the same input artifact from several sources.

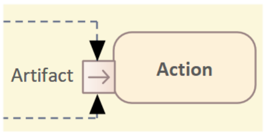


Figure 3.14 Receiving same input artifact from several sources

3.6 Guidelines for Types, Names, Visibility

In general, it is beneficial to use good, telling names, and to avoid the cluttering of diagrams with redundant text.

23) Method Artefact types and names.

If you create a Method Artifact (during the Method Specification), you give it a name in the *"Element"* tab of its properties. This name is then shown as *Type* in the *"ActivityParameter"* section of the *"Element"* tab in the Properties. In the workflow diagram, it appears as *": typename"*.

At the top of that Element tab, there is a *Name* entry. You can leave it empty or put a special string there. It then appears as *"name :typename"* in the diagram. It is suggested to always use only the type, unless an additional differentiation is needed.

24) Activity Artefact types and names.

Whenever you create an Activity Artifact (by drawing *"Exposed artifact on activity"* from the toolbox onto an Activity), the respective pin has the default name *ArtivityArtifact* (in the *"Element"* tab of its Properties). It is recommended to not use this name; instead, you can delete it or give it a telling name.

Additionally, you can give each pin a type. This can be done in the *"Pin"* tab (or in the *"ActivityParameter"* section of the *"Element"* tab, if present), within the *"Properties"* section, by selecting one of the method artefacts. Name and type appear as *"name: type"* in the diagram.

It is up to you to use it or not. Feasible uses of both name and type are, e.g.:

- distinguish between several subtypes of a type, e.g.,*"Security: report"* and *"Performance: report"*.
- distinguish between different artefact versions: *"prelim: Model"* and *"final: Model"*.

25) Avoid misleading characters in labels.

When labelling an element, avoid characters such as ".", ":", "[ ]", "::", "<< >>", since they can be misinterpreted as typing or stereotyping. Rather prefer "-" or "_" instead of blanks.

26) Visibility of Labels and Stereotypes.

Avoid cluttering diagrams with names. Hence, use name labels only when not clear from other labels, in particular for artifact pins. Furthermore, see the FAQ section for an explanation on how to hide stereotype names in diagrams. In particular, to show which artifacts are transmitted between actions, label the Artifact flow between two pins rather than the pins themselves. Furthermore, Artifact flows from method input / to output MethodArtifacts need not to be labelled if the MethodArtifacts are labelled (and not too far away). Figure 3.15 shows examples on how to label the artefact flow.

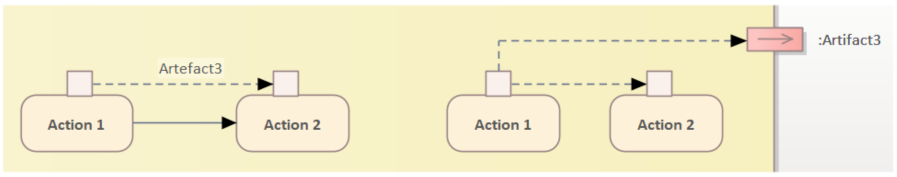


Figure 3.15 Labeling artifact flow

27) Layout Control flows from top/left to right/bottom.

We suggest to place the *"Start workflow"* symbol close to left or top edge of diagram and to let the control flow evolve towards right/bottom.

28) General Appearance of Diagrams.

In general, make the whole diagram as readable as possible. For instance, avoid unnecessary crosscutting of flows, and position their start and endpoints on actions clearly/unambiguously.

29) Usage of "Boundaries" for grouping.

UML allows free use of the so-called boundaries, typically hollow boxes with solid border. They can be used to indicate grouping of some elements within a workflow diagram.

In general, their use is not recommended in VVML; instead, putting the grouped elements into an own method is preferred. However, if this appears inadequate, use dotted lines and, perhaps, a color other than black if it is possible. Figure 3.16 shows an example of boundaries for grouping parts of a workflow.

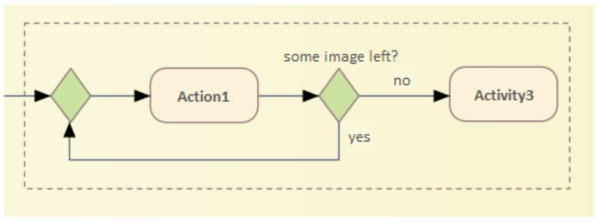


Figure 3.16 Boundaries for grouping

Chapter 4 Behind the Scenes

This chapter describes the mechanisms implemented in EA for VVML.

4.1 VVML Metamodel (UML profile)

The VVML profile, depicted in Figure 4.1, is a set of stereotypes extending the UML Metaclasses used for behavioural Activity diagram, which is too complex to be used off-the-shelf without any simplification, by some V&V specific elements. A Profile is a collection of extensions, based on stereotypes that are applied to UML elements, connectors, and features, as shown in Figure 4.1.

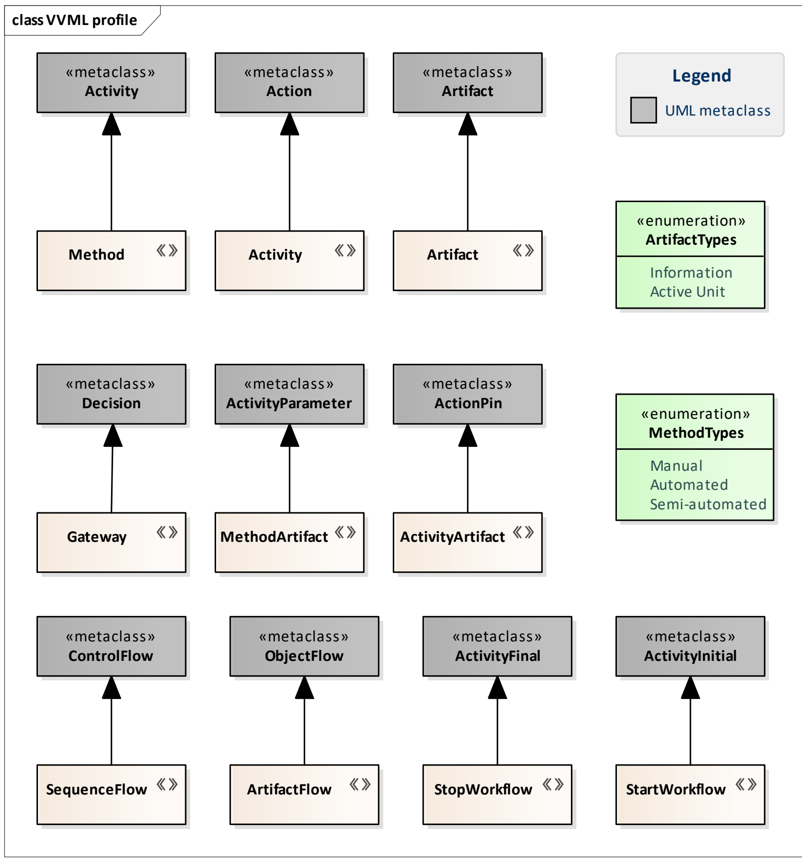


Figure 4.1 Metamodel diagram illustration the extension of the core UML Activity diagram elements for the sake of VVML DSL

4.2 DSL Metamodel Details

Figure 4.2 shows the UML-based DSL metamodel, defining a set of rules, principles and conventions of constructing a model for creating a VVML method workflow diagram.

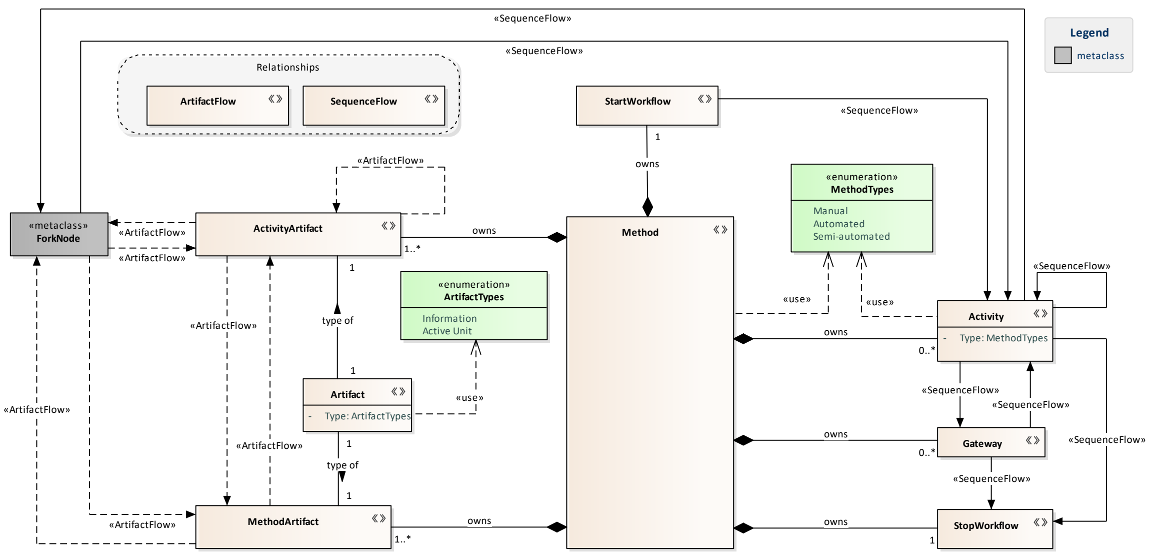


Figure 4.2 DSL for Method workflow

Main elements for specifying workflows are:

- VVML stereotypes «Activity», «StartWorkflow», «StopWorkflow», «ActivityArtifact», «ArtifactFlow» and «SequenceFlow»

Optional elements for specifying workflows are:

- VVML stereotypes «Gateway», «Activity» type *callBehavior*
- UML Fork / Join

Following Figure 4.3 depicts a subset of Figure 4.2 concerning Method definition.

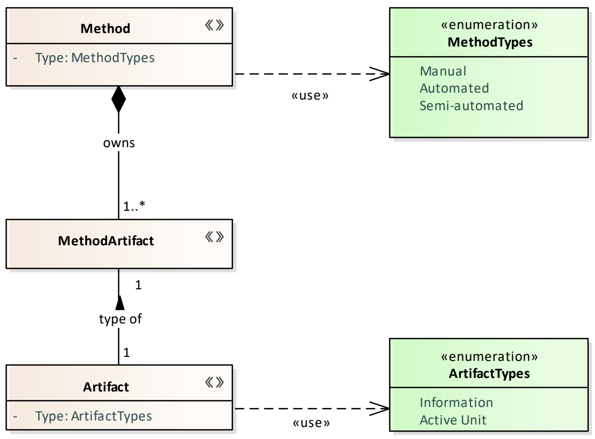


Figure 4.3 DSL for Method definition

Chapter 5 Frequently Asked Questions (FAQs)

This chapter contains common practical questions related to the usage of the VVML tools in EA.

5.1 What is the meaning of those funny symbols that appear on the right side of an item?

When you click on an element in the workflow diagram, several things happen: (i) the item is emphasized by a frame, (ii) the EA Properties window shows its properties, and (iii) some symbols appear to its right side (one is shown in Figure 5.1).

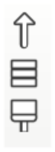


Figure 5.1 EA Symbols when clicking on diagrams.

For VVML, the arrow is helpful: left-clicking on it, you can directly draw an appropriate arrow from it to another item in the diagram.

5.2 How can I switch between Workflow diagram and Method diagram?

If you are editing a Workflow diagram and you want to come back to the Method diagram, you can follow two ways:

- You can double-click on the diagram entry, directly under the respective package entry, in the Browser window. Then, this diagram is also opened in the EA editing area.
- You can click on the ◎ Symbol in the respective tab header. This choice is recommended.

If there are changes to be saved, which is indicated by an asterisk before the method name (see Figure 5.2), you will be asked whether to save them or dismiss them.

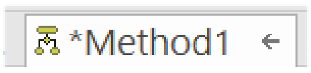


Figure 5.2 Diagram tab in Enterprise Architect indicating unsaved changes.

After that, the tab content will be replaced by that of the method definition.

5.3 How to save changes?

Changes in a diagram tab are always indicated by the asterisk shown above (Figure 5.2). Right-click on the tab name opens the context menu, where you can select *"Save changes to …"*.

5.4 How to find an EA window? (For instance, *"Features"*)

Open the top EA menu Start and here Design, or top menu Design, and here Editors. Figure 5.3 shows the menu icons in EA. This will open a list where you can select the wanted EA window.

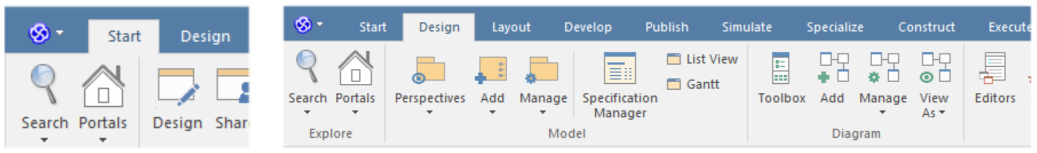
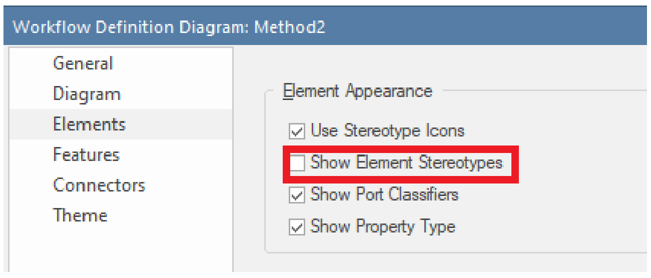


Figure 5.3 EA menu icons

5.5 How can I suppress stereotypes on a diagram?

1. How to suppress Element stereotypes in a diagram.
   1.1. It is possible to show a number of element characteristics on a selected diagram. You can define which of these characteristics to show, by right-clicking the corresponding checkboxes on the *"Elements"* tab of the diagram, *"Properties"* dialog (and, conversely, which to hide by clearing their checkboxes). Access from the context menu by a right-click background of open diagram | Properties | Elements and deselect the checkbox to hide element stereotypes highlighted in the figure below. Please note that this works only on method diagrams, but not on workflow diagrams. Figure 5.4 shows the configuration dialog for elements in workflow diagrams.



Workflow Definition Diagram: Method2

General
Diagram
Elements
Features
Connectors
Theme

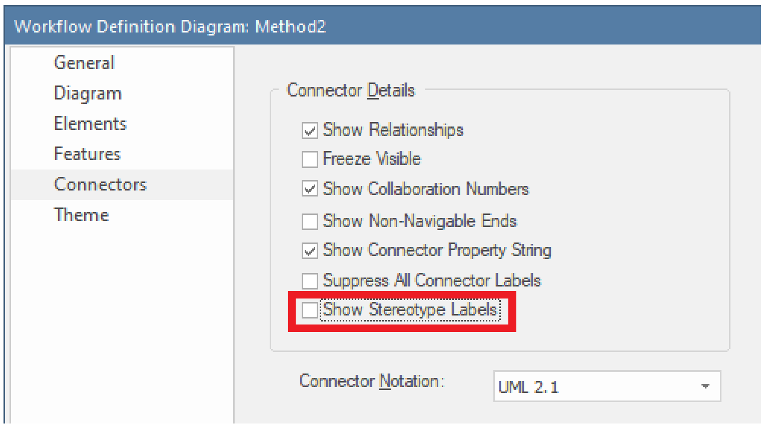
Element Appearance

- [✓] Use Stereotype Icons
- [ ] Show Element Stereotypes
- [✓] Show Port Classifiers
- [✓] Show Property Type

Figure 5.4 Workflow diagram configuration dialog: Element Appearance

2. How to suppress stereotype labels of the connectors in a diagram.
   2.1. It is possible to configure a number of aspects related to how connectors are shown on a diagram, using the *"Connectors"* tab of the diagram, and the *"Properties"* dialog. These settings can be customized from a context menu reached by right-clicking the background of an open diagram and selecting *"Properties"*. To suppress the stereotype labels, navigate to section *"Connectors"* and deselect the checkbox *"Show Stereotype Labels"* (as shown in

Figure 5.5). Please note that this method works for method diagrams but does not work for workflow diagrams.



Workflow Definition Diagram: Method2

- General
- Diagram
- Elements
- Features
- Connectors
- Theme

Connector Details

- [✓] Show Relationships
- [ ] Freeze Visible
- [✓] Show Collaboration Numbers
- [ ] Show Non-Navigable Ends
- [✓] Show Connector Property String
- [ ] Suppress All Connector Labels
- [ ] Show Stereotype Labels

Connector Notation: UML 2.1

Figure 5.5 Workflow diagram configuration dialog: Connector Details

References

[1] Sparx systems, ”Enterprise Architect,” Sparx systems, [Online]. Available: https://sparxsystems.com/enterprise_architect_user_guide. [Accessed 29 05 2023].

VALU3S

www.valu3s.eu

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.