Contents

1 The Perspective **1**
- 1.1 VSDT Editor Views . . . 1
- 1.2 General-purpose Eclipse Views . . . 2
- 1.3 Additional Views for the VSDT . . . 3

2 Basic Tutorial **7**
- 2.1 Creating a new Business Process Diagram . . . 7
- 2.2 Setting up Participants and Business Processes . . . 7
- 2.3 Modelling a basic Business Process . . . 8
- 2.4 In-depth Modelling . . . 9
- 2.5 Validation and Simulation . . . 11
- 2.6 Code Generation . . . 11

3 Selected Features **13**
- 3.1 GMF Modelling Assistance . . . 13
- 3.2 Managing Non-Visual Elements . . . 13
- 3.3 Expressions . . . 15
- 3.4 Service Parameter Assignments . . . 15
- 3.5 Inserting Elements and Patterns . . . 16
- 3.6 Appending Flow Objects . . . 17
- 3.7 Connecting Flow Objects to a Sequence . . . 17
- 3.8 Model Validation . . . 17
- 3.9 Text Generation . . . 18
- 3.10 Simulation and Interpretation . . . 18
- 3.11 Import and Merging of Process Diagrams . . . 19
- 3.12 Structure-Based Layout . . . 19

4 JIAC Node Plugin **21**

Bibliography **23**

1 The Perspective

This section will briefly introduce those parts of the Eclipse GUI that are relevant for the work with the Visual Service Design Tool. These views are aggregated in the Easy Service Creation perspective, which an be selected via the Menu *Window* → *Perspective*. Figure **??** (page **??**) is showing a screenshot of the Visual Service Design Tool featuring most of the relevant views. In the following each of the views will be introduced briefly.

1.1 VSDT Editor Views

The editor window is shown automatically when opening a file in the Navigator view. Depending on the PlugIns currently installed this can be a plain text editor, a browser, an elaborate code editor or some sort of graphical editor. For the Visual Service Design Tool there are two editors available: A visual editor showing the BPMN graph and a tree editor reflecting the internal structure.

The Graphical Editors These are the primary editors when working with the VSDT (see Figure 1.1):

- The *Business Process System* editor is opened when the diagram file is clicked. It is used for organizing the several interdependent Business Processes which make up the system as a whole, as well as the Participants involved in these Processes.

- The *Business Process Diagram* editor is opened when double-clicking one of the Business Process Diagram nodes in the Business Process System editor. This editor is the actual BPMN editor used for modelling the individual Business Processes.

Both editors feature a palette with the nodes and connections. For placing a node on the canvas or inside a compartment of another node (e.g. a Pool or a Sub Process) click the icon in the palette and then click again on the canvas. For drawing connections, click on the first node, draw the connection to the second node and release the mouse button. Note that nodes and connections can not be drawn arbitrarily, but have to follow the BPMN syntax, e.g. a Task can only be drawn inside a Pool and a Sequence Flow can only connect Flow Object within the same Pool.

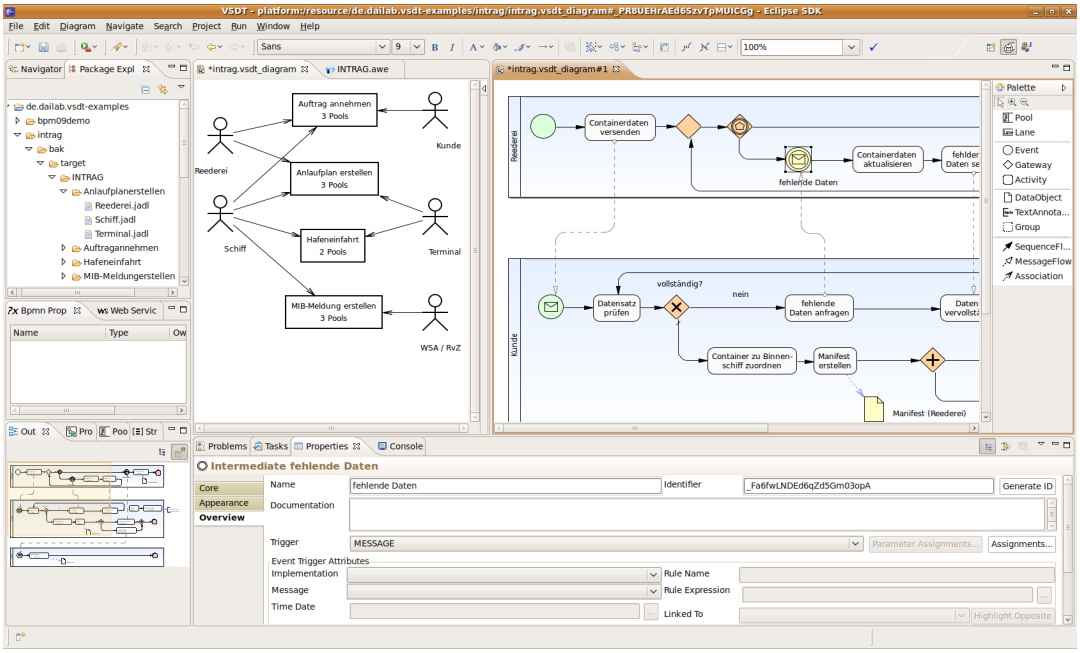

Figure 1.1: Business Process System and BPMN editor shown side-by-side.

The Tree Editor The tree editor can be useful for managing and editing those parts of the Business Process Diagram that do not have a graphical representation.[1] Note that the tree editor is more powerful than the graphical editor, and the diagram might be invalidated when doing certain operations in the tree editor. Especially, the tree editor should *not* be used for creating any elements that *do* have a graphical representation in the diagram, as this representation will not be created along with the element.

The Text Editor Both the diagram and the model file can be opened with any text editor and edited as XML. While this can be helpful for adapting existing models to changes in the metamodel of a newer version of the VSDT, you should in general avoid editing the files XML sources, as this can render them unreadable for the other editors.

1.2 General-purpose Eclipse Views

In the following those standard views of the Eclipse IDE will be introduced, that are relevant for the work with the VSDT.

[1] Although in general it is not necessary to use the tree editor, as the graphical editor provides means for editing non-graphical elements, too.

The Project Explorer Here the user can manage his projects and create and delete files. Note that Eclipse provides different similar views for managing files, e.g. the Project Explorer, Navigator, or the Package Explorer, each providing slightly different features.

The Properties View Although some attributes, like an element's name, can be edited in the graphical editor view as well, for most other attributes the properties view will be needed, where all the attributes relevant to the user can be inspected and edited. Of course, each change done in the properties view can be undone and redone and the editor will be immediately updated. There are two tabs available in the properties view: The *Core* tab provides a table showing the attributes in categories and in alphabetic order. The *Overview* tab provides a clearer look, grouping the attributes and arranging them by relevance in two columns. Additionally, the Outline tab features a number of buttons, providing access to additional dialogs for managing e.g. an Activity's Properties and Assignments.

The Outline This view provides a short outline of the current editor's content. In case of a graphical editor, like the VSDT, this can be a miniature view of the entire diagram, and in case of a tree editor an additional tree view for easier navigation.

The Problem View This view lists all the problems that have been found in the model, subdivided in errors and warnings. By double-clicking one of the items the editor will focus on the element the problem occurred on (for refreshing the errors shown in the Problem View, select *Diagram — Validate* from the menu).

The Error Log Other than the Problem View, the Error Log will log problems with the editor itself. So if you encounter strange behaviour or in case the editor should crash you can check here for the reason and send in an error report.

1.3 Additional Views for the VSDT

The following are views that have been crafted especially for the VSDT.

The BPMN-Properties View The BPMN-Properties View (see Figure 1.2, to the left) provides an easy way to inspect the Properties in the scope of the currently selected element in the active editor, i.e. the Properties that can be used in an Assignment owned by that element. The property scope of a BPMN element comprises the Properties of (a) that element itself, e.g. an Activity, (b) Messages going in and out of that element, e.g. the in- and output parameters of a Web service call, and (c) the (transitive) parents of that element, e.g. (Sub-) Processes.

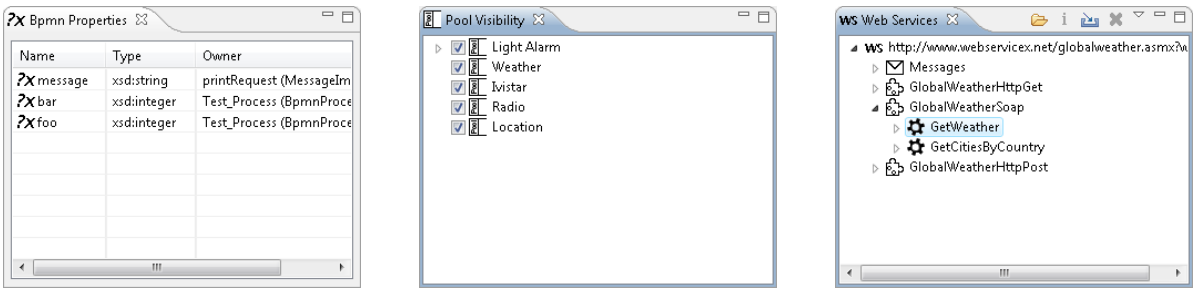

Figure 1.2: The BPMN Properties View, Pool Visibility View, and Web Service View.

The Properties are displayed in three columns, showing the name and the type of the Property and the name and the element type of the Property's parent element. The properties an be sorted by clicking on the column heads. By double-clicking on a Property, an Organize Properties Dialog will be opened for the Property's parent element.

The Pool Visibility View In the Pool Visibility View, which is seen in the centre of Figure 1.2, all the Pools in the diagram are displayed. If the check box in front of an entry is unchecked, the corresponding Pool and all incoming and outgoing connections, e.g. Message Flows, will be hidden. This feature can be of some use in diagrams holding many Pools: When modelling three or more interconnected Pools, Message Flows going from the first to the third Pool might cross the second Pool, which can be confusing when editing that Pool. In this case, the first or the third Pool may be hidden, so the Message Flows (which are then hidden, too) do not longer obstruct the view on the second Pool. In the same way the Pools and Message Flows can be shown again by checking the corresponding check box. Note that these settings are not persisted, so when closing and re-opening a diagram all Pools will be visible again.

The Web Service View The Web Service View (see Figure 1.2, to the right) provides access to Web Services, which can be inspected and imported into the currently opened diagram. Web Services can be added to the list by clicking the Open button and entering the exact URL of the WSDL definitions file. The Web Service is displayed as a tree, including the various Messages and their types, nad the Port Types, their Operations, and their In- and Output Messages. By clicking the Info button, the complete WSDL definition is shown in plain XML. Most importantly, Messages and Operations can be imported from the Web service description into the Business Process Diagrams, so they can be reused in a Web service invocation.

The Interpreter View BPMN diagrams created with the VSDT can be simulated and interpreted using the built-in process interpreter (see Figure 1.3 and Section 3.10). For

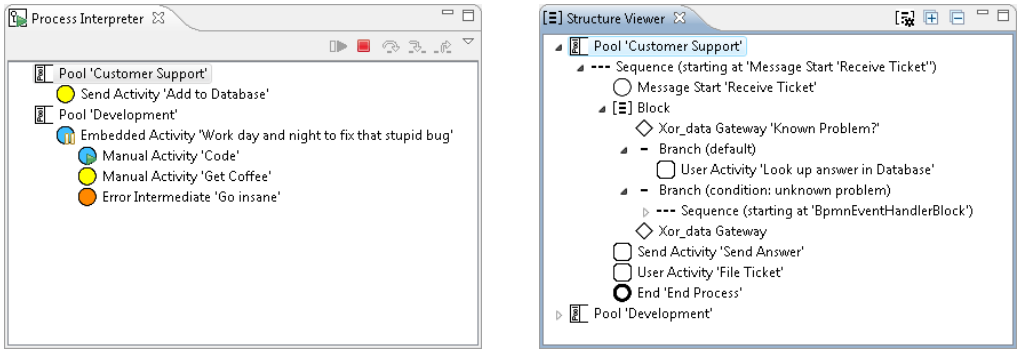

Figure 1.3: The Interpreter View and Structure View.

starting a simulation, first switch to the BPMN diagram you want to simulate. Open the Process Interpreter View and click the *Start* button. For each Pool in the diagram, the view will show those Activities that are currently ACTIVE or READY. For advancing a step in the simulation, expand a Pool and double-click one of the listed elements, that is the Flow Objects currently being ready, e.g. Start Events. For more control, you can also select one of *Step Over*, *Step Into* or *Step Out*. Hit the Stop button for ending the simulation and removing the markers from the diagram editor view.

The Structure View This view allows to apply the Structure Mapping (see **??**) at modelling time, displaying the results in a tree. By clicking on an element, the corresponding node in the diagram is highlighted. Further, the user is notified if there seem to be structural conflicts in the process, which is determined from the number of left-over Sequence Flows. This view should prove highly useful for validating the structure of processes prior to transformation to executable code (see Figure 1.3).

2 Basic Tutorial

In this chapter the user will be guided through the creation of a simple Business Process Diagram, from creating the diagram file to validation and code generation.

2.1 Creating a new Business Process Diagram

These are the basic steps for creating your first Business Process Diagram:

1. Start Eclipse and select a location for the workspace. This is where all the projects – BPMN and others – will be stored. When starting Eclipse for the first time, a welcome screen will be shown. Read something about the features of Eclipse, if you want, then exit the screen.

2. Change to the *VSDT* Perspective.

3. Select *New* → *Project* → *General* → *Project* in the menu bar. Open the Navigator View to see the newly created Project.

4. On the project, select *New* → *VSDT Meta Diagram* and enter a name for the file. On the last page of the wizard some of the global settings for the Business Process Diagram, such as the Title, Description and Author name, can be set. A file with the extension 'vsdtd' is created, holding both the semantic model and the notational model (i.e. the layout information).

   Note that the files are stored in XML format and can be edited with a text editor, too. However, you should do so only to fix a broken file.

5. By now, the diagram should have opened automatically; otherwise open it manually by double-clicking it. It will be opened with the graphical editor.

2.2 Setting up Participants and Business Processes

With the VSDT, not only individual Business Process Diagrams, but sets of Business Process Diagrams belonging to the same scenario – here referred to as Business Process Systems – can be modelled. For this, the modelling starts with defining the several Participants and the Business Processes they participate in.

1. Select a **Participant** from the palette and click on the canvas. A stick-figure will appear. Repeat for each Participant relevant for the Business Process System. These can be companies, roles, computer systems or individual persons.

2. Now select **Business Process Diagram** from the palette and draw it on the canvas. These represent the individual processes (quite similar to 'Use cases').

3. Now select the connection form the palette and connect the Participants with the Business Processes.

4. Finally, perform a double-click on one of the Business Process Diagram nodes, which will open it in a new diagram editor.

2.3 Modelling a basic Business Process

Next, we will formulate a simple business process. Here, we will focus on the visual elements of BPMN.

1. To get started, perform a right-click on the canvas and select *Initialize* → *Initialize Pools* and hit *OK* to confirm the dialog. For each Participant associated with the Business Process a Pool will be created.

2. Alternatively, select **Pool** from the top of the palette and move the mouse to the canvas. Push the mouse button and drag it to the lower right to create a large Pool. Enter a name for the Pool and select one of the Participants associated with the Business Process using the Properties view.

3. Along with the Pool also a Lane will be created. To create more Lanes, select the **Lane** element from the palette and click on the Pool's label (as existing Lanes will fill the Pool's compartment completely). Note that the Lanes can not be moved manually. According to the BPMN specification the first Lane will be invisible (faded out in the editor).

4. Let's create some **Flow Objects** inside of the Lane. Select one of the Flow Object from the palette, i.e. Events, Activities and Gateways, and click inside of the Lane. In case you selected the Event, a small menu will appear, asking whether to create a Start, End, or Intermediate Event; otherwise the element will be created right away.

5. Select the **Sequence Flow** icon from the palette and connect the several Flow Objects by pressing the mouse button on the source and dragging it to the target. When connecting the Activity be sure to aim for the label. If you hit the Activity's compartment you can not create a connection. You can change the routing style from the toolbar or add more bendpoints to a connection by dragging it.

6. Use the Property Sheets to alter the Elements' name, description, type, and type-specific attributes. Select the element, e.g. an event, and open Eclipse's Property View. Select the Overview sheet from the tabs to the left to find a clearly arranged form holding the various attributes. If you want to set only the type of a Flow Object, e.g. for making an Event a *Message* Event, you can also use the element's context menu and select *Edit...* → *Set Type*, or use the keyboard shortcut `Ctrl+T`.

7. Now select the **Message Flow** icon from the palette. Select an Activity or an End Event as source and draw the Message Flow to an element in a different Pool, or to some point beneath the Pool and select to create a new Pool element there.

8. Finally, we will associate an Activity with a **Data Object** (however, this will not affect the generated BPEL code). Select the Data Object from the palette and create it on the canvas. Select the Association connection from the palette and connect the Data Object to the Activity. Select *BPMN* → *Initialize Input/Output Set* from the Activity's context menu, depending on the associations direction. Notice the new Input Element in the Activity's property sheet. This Input Set references all the Activity's incoming/outgoing Data Objects.

2.4 In-depth Modelling

Now that we created the diagram visuals, this section deals with the equally important underlying, non-visual parts of BPMN, such as properties, assignments, conditions, and service invocations.

1. Right-click the Message Flow or open its property sheet and select *Initialize Message*. Note that by doing so the End Event's type changes to `Message`. A new (non-visual) Message object has been created and associated with the Message Flow, and its source and target, if possible.

2. To define a (Web-) service invocation, select the *Organize Implementations* and *Organize Messages* buttons from the Business Process Diagram's property sheet. Select the newly created Messages and Implementations (services) from the list and set the values according to the service to invoke. Alternatively, Web services can also be imported using the Web Service View, which is much more comfortable and will be explained in depth later.

3. Next, we will define the process data, i.e. Properties associated to the Pool's Process. Open the Pool's overview property sheet and click *Organize Process Properties* or select the respective item from the Pool's context menu. Create some properties using the buttons in the shown dialog and edit the values of the selected Property using the text fields in the lower part of the dialog. Besides

the top-level process, Tasks and Subprocesses can hold Properties, too, which are available only for that activity or its child activities, if any.

4. To assign a value to a Property, you have to create an Assignment. Open the properly sheet of some element in the process and click the Assignments button. Create a new Assignment, select the Property and enter an Expression. Click the button with the dots (...) on it to open another dialog helping you to enter and validate an expression using the VSDT Expression Language VXL (see Appendix **??**).

5. Now that the Properties are declared and assigned a value, they can be used e.g. in condition expressions. Select a Sequence Flow coming from a Gateway (a point where the flow of control branches), set the Condition Type to *Expression* and enter the Condition Expression. Again, use the button with the three dots(...) to validate the Expression.

6. Just like Processes and Activities, Messages can have Properties. Again, the dialog can be accessed via the property sheet or the context menu. Add some properties to the newly created Message(s), being the input and output parameters of the respective Web service.

7. To pass the parameter values to the Message, create one or more Assignments on the Flow Object the Message is going in or out of. There are two ways for doing this:

   - The easiest way is to use the *Parameter Assignment Dialog*. Select the Activity or Event sending or receiving the Message(s) and hit the respective button in its property sheet. The dialog will show all of the messages' input and output Properties and offer drop-down menus for selecting another Property or entering an individual Expression to be assigned to these parameters.
   - For more control over the parameter assignments, you can also open the *Organize Assignment Dialog* via the Flow Object's property sheet or context menu and manually create the individual Assignments. Select a Property to assign the value to, e.g. one of the input parameters of the Web service's input message, and enter a from expression.

   To refer to a Property in the expression, just type in the Property's name with a leading `$`, e.g. `$foo + 1`. Note the assign time value: if this is set to before, the assignment will be made before the Activity is executed, i.e. the Web service is invoked, otherwise the assignment will be made afterwards. Thus this value should be set to *before*, when passing values from the process to a Web service's input parameter, and to *after*, when passing values from the Web service's output parameter back to the process.

2.5 Validation and Simulation

When your process is done – or seems to be done – you should validate it. There are several means for validation in the VSDT: First, you can validate the process diagram against the constraints given in the BPMN specification; second, you can check the structure of the process, which is important for most transformations to executable code; and third, you can run a simulation, testing the several Expressions, Conditions, and Assignments.

1. To validate the diagram against the constraints from the BPMN specification, select *Diagram* → *Validate* from the menu, or by clicking the checkmark symbol in the tool bar. You might notice some error or warning marks in your diagram or entries in the problem view. You should fix these problems before exporting the diagram to executable code and validate the diagram again.

2. For checking the structure of the process, open the Structure View (see Section 1.3) and click the *Structurize* button. This will trigger the same Structure Mapping used in the actual transformations and display the result, i.e. a structured form of the process, featuring elements such as sequences and blocks. While the structured model might be a bit cumbersome to read, it gives evidence of the structure that will be recognized from the process, and if this is not the structure you intended you should consider restructuring the process. Unfortunately, most executable languages are much more restrictive than process notations such as BPMN, so this check is necessary.

3. For a more in-depth validation of the process you may consider running a simulation. Currently there are two types of simulations implemented: a manual simulation and an interpreting simulation (see Section 3.10).

   - Use the manual simulation to get a feel of how the process behaves when taking a certain path, and to identify possible deadlock situations
   - When using the VSDT Expression Language (VXL, see Section 3.3), the interpreting simulation will help you validating the several condition and assignment expressions used throughout the process.

2.6 Code Generation

Once all three validations are successful, the diagram can be translated to executable code. Of course, there might still be semantic errors in the process the validation can not uncover, so you should think about thoroughly testing the resulting program code before deploying it to productive use.

1. Once the diagram shows no more errors, it can be exported to executable code. Select *Export...* from the file menu or from the model file's context menu. Select the desired target language from the *BPMN Export* group and proceed through the dialog. Select the model file(s) to be exported, adjust the target directory or the other options, if necessary, and hit the *Finish* button.

2. The export might take some seconds. If the model is sound, the output files will be created in a new directory in the specified target directory, named after the Business Process Diagram. By default, also a log file will be created along with the model files in the directory *exportLogs* in the specified target directory.

3. If the process has been modelled accurately, the resulting program can be readily executable. Still it is recommended to check the result with a native editor for the respective language, to be sure the files are free from defects.

3 Selected Features

This chapter will give further insight on how to use some of the features of the Visual Service Design Tool.

3.1 GMF Modelling Assistance

The Eclipse Graphical Modelling framework the VSDT is based upon comes with a number of valuable modelling assistance features (if not desired, modeling assistance can be turned off in the preferences). In the following some of these will be briefly introduced. Figure 3.1 is showing the modelling assistance in use.

- When resting the mouse on top of a compartment, a small palette will show up, showing the elements that can be placed in this compartment. Thus one does not have to go all the way back to the palette for creating a new node.
- When resting the mouse on top of a node, small arrows going in and out of the node will appear. By dragging these arrows new connections can be drawn.[1]
- When a connection is drawn and the mouse button is released over the canvas or another compartment, a node can be created in that place along with the connection.
- In case multiple node- or connection-types can be created using a given tool in a given context, the user will be prompted to select one.

3.2 Managing Non-Visual Elements

For each of the non-visual elements — Properties, Assignments, Messages, and Implementations— a management dialog has been written. The dialogs follow a clear and recognizable layout, showing the elements as-is in a list along with a number of buttons for inserting, removing and sorting of the elements and text fields for editing the attributes of the currently selected item (see Figure 3.2).

The various dialogs can be accessed in the following ways:

[1] Depending on the location, different connections will be offered: In the top and bottom region of a node incoming and outgoing Message Flows, in the left half incoming Sequence Flows and Associations, and in the right half outgoing Sequence Flows and Associations respectively.

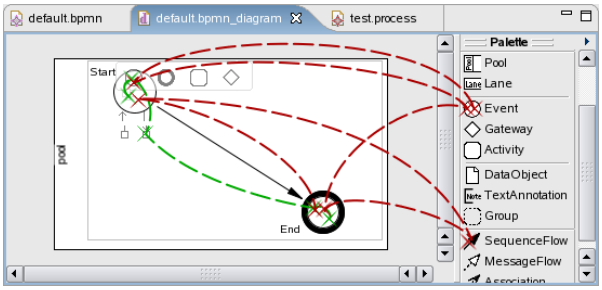

Figure 3.1: Mouse movement with and without the use of the Modelling Assistant.

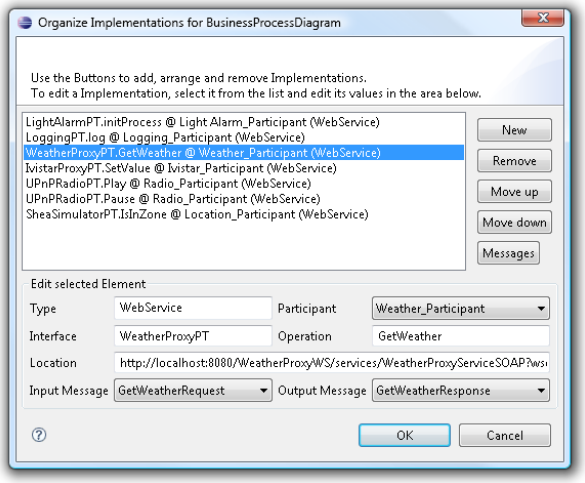

Figure 3.2: A Supporting Type Organization Dialog.

- The Organize Properties Dialog can be accessed via the context menu and property sheet of Pools, Activities and Message Flows, and by double-clicking an element in the BPMN-Properties View or a Message Flow.[2]

- The Organize Assignments Dialog can be accessed via the context menu and property sheet of Pools and Flow Objects, and by double-clicking Flow Objects.

- The Organize Messages Dialog can be accessed via the context menu and property sheet of the Business Process Diagram and the property sheet of Message Flows.

- The Organize Implementations Dialog can be accessed via the context menu and property sheet of the Business Process Diagram.

[2]In the case of Message Flows, the Properties of the underlying Message, if any, will be edited, and in the case of Pools, the Properties of the Pool's Process.

3.3 Expressions

The BPMN standard does not specify an expression language to be used. Instead, it is assumed that the language of the target framework is used, e.g. XPath. However, in a tool that provides transformations to various target frameworks this is not an option. While the diagram structure could be translated to the syntax of the target system, the expression, given that they are written in an unknown language, could not – although all those languages might be very similar. To address this flaw, the VSDT comes with its own, very simple expression language, the *VSDT Expression Language*, or VXL for short. The advantage of using VXL is that it provides a greatest common divisor of the expression languages used in the target frameworks. Thus, most expressions can be given using VXL, in which case they can be validated and – more importantly – parsed and translated to the respective expression languages used in the target frameworks.

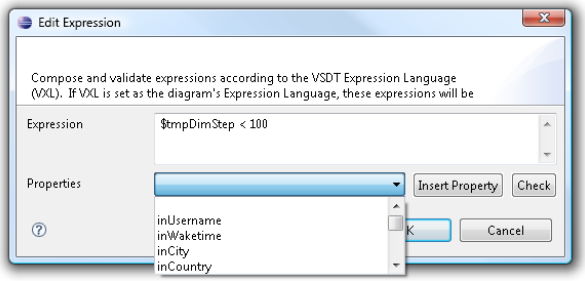

Figure 3.3: The Edit Expression Dialog.

Each text field referring to an Expression in the dialogs and property sheets of the VSDT provides a small button for opening the Edit Expression Dialog, which can be seen in Figure 3.3. This dialog not only provides a large text field for editing the Expression, but also a list of all Properties visible in the scope of the element owning the Expression, which can be selected from the list and inserted into the expression. Further, the *Check* button can be used to validate the Expression, which will check both the syntax and the availability of the variables used in the expression.

Note that there is no type checking yet. However, this feature is on the agenda, and will be implemented as soon as possible.

3.4 Service Parameter Assignments

While the *Organize Assignments Dialog* provides means for organizing all types of Assignments, it can be quite weary to make the assignments to a service call, passing a number of values to the service's input parameters and storing its results in other local variables. Moreover, this is a common source for mistakes, like selecting the wrong assign time, or missing an important input parameter. Using the *Parameter Assignments*

Dialog this task can be facilitated in many ways.

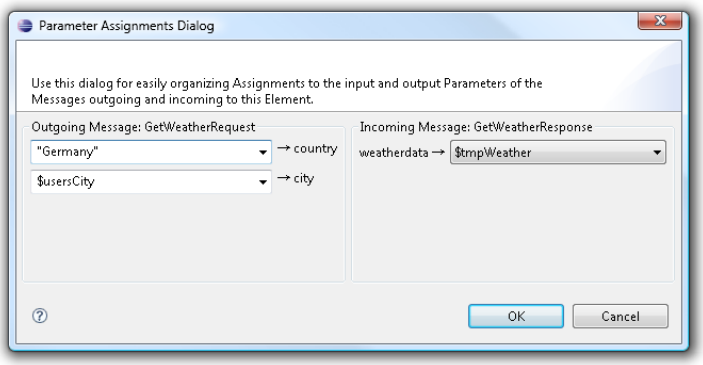

Figure 3.4: Parameter Assignments Dialog

This dialog is available for all Activities and Events sending or receiving messages, such as the Message Event and Send, Receive, Service and User Activity. Provided that the Implementation and the input and output Messages for that service are specified, the dialog displays a drop-down menu for each of the incoming and/or outgoing Messages' Properties. The values set in these lists will then be used for the Assignments to the input and output parameters. For the outgoing message, arbitrary expressions can be inserted, while the parameters of the incoming message can only be assigned to a local Property. If more specific Assignments are needed, the dialog can still be used for generating stubs for those Assignments, which then can be refined in the *Organize Assignments Dialog*. Further, the dialog will notify the user if there are input parameters that have no value assigned to them.

3.5 Inserting Elements and Patterns

By right-clicking on a Sequence Flow the *Insert...* menu can be reached. Here it is possible to insert a new element in between the source and the target of that Sequence Flow, which is very useful for extending existing diagrams. The existing sequence Flow will be reoriented to the new element, preserving existing attributes such as the condition, and a second Sequence Flow is drawn from the new element to the existing Sequence Flow's former target.

Apart from simple elements such as Activities, Intermediate Events and Gateways, it is also possible to insert complex workflow patterns, such as a split/merge block or a loop. This does not only greatly reduce the time needed for the diagram creation[3], but also ensures that the workflow is correct (Referred to as "correctness by construction").

[3]Reducing the pure editing time by up to 70% according to [1].

Note that by now the layout of the diagram will not be adapted to the newly inserted elements, thus the user will have to rearrange the surrounding elements to make room for the new nodes.

3.6 Appending Flow Objects

Similarly to the *Insert* Actions, the *Append* Actions can be used for quickly appending new Flow Objects after existing ones. It can be reached through a Flow Object's context menu, or – more conveniently – by using the keyboard shortcuts `Ctrl+Shift+(A|G|I|E)` for appending Activities, Gateways, Intermediate and End Events. Thus, after the first Start Event has been placed, the basic workflow can quickly be assembled using only the Append action and the TAB key to navigate between the existing nodes.

3.7 Connecting Flow Objects to a Sequence

Further, a group of Flow Objects can be selected and connected with Sequence Flows using *Connect to Sequence* Action or the keyboard shortcut `Ctrl+Shift+C`. The Flow Objects will be connected in the order they have been selected. Therefore they should be selected one by one (holding down the Shift or Ctrl key), and not using a selection margin.

3.8 Model Validation

VSDT provides two sorts of validation: Validation of BPMN constraints, and structural validation.

Validation of Constraints BPMN diagrams can be checked to conform to the constraints given in the BPMN specification by selecting *Diagram* → *Validate* from the menu or by clicking the checkmark icon in the tool bar. Afterwards errors will be listed in the problem view. Additionally, faulty or otherwise problematic elements will be marked with a respective icon in the process graph.

Structural Validation Besides the individual elements, also the structures in which these elements are connected are important. For checking the structure of the process, open the Structure View (see Section 1.3) and click the *Structurize* button. This will trigger the same Structure Mapping used in the actual transformations and display the result, i.e. a structured form of the process, featuring elements such as sequences and blocks. While the structured model might be a bit cumbersome to read, it gives evidence of the structure that will be recognized from the process, and if this is not the structure you intended you should consider restructuring the process.

3.9 Text Generation

The VSDT features a powerful transformation of the Business Process Diagram to natural language text. Currently only English text is supported, but other languages may be included in the future, as well. The output text can have different formats, e.g. plain text, HTML or Latex, which to use can be selected in the Export Wizard. While this feature is yet at an early stage, it can already be used for quickly generating documentation for those who can not read the process diagrams or for media where they are difficult to present, e.g. in a talk. Emphasis has been laid on preserving the process structure as much as possible in the text, e.g. using indentation. Further a number of randomly selected redundant terms is used to increase the linguistic diversity of the resulting text.

3.10 Simulation and Interpretation

BPMN diagrams created with the VSDT can be simulated and interpreted using the built-in process interpreter (see Section 1.3). For starting a simulation, first switch to the BPMN diagram you want to simulate. Open the *Process Interpreter* view and click the *Start* button. For each Pool in the diagram, the view will show those Activities that are currently ACTIVE or READY. For advancing a step in the simulation, expand a Pool and double-click one of the listed elements, that is the Flow Objects currently being ready, e.g. Start Events. For more control, you can also select one of *Step Over*, *Step Into* or *Step Out*. Hit the Stop button for ending the simulation and removing the markers from the diagram editor view. In the diagram itself, the Flow Objects are annotated with a marker symbol representing their state (see Table 3.1).

Table 3.1: Mapping of Marker Colors to Flow Object States

Yellow	Ready for execution
Blue	Currently active / executing
Green	Executed succesfully
Red	Execution failed or interrupted
None	Not yet executed or ready; idle

Once the simulation is running, the user can *Step Over*, *Step Into* and *Step Out* of Flow Objects. Stepping into a Flow Object is particularly interesting for Activities with attached Event Handlers or Embedded Subprocesses, for which it is the default behaviour. Different kinds of interpretations are available (or planned for the future):

- *Manual Simulation*: Here, the user is asked which way to proceed when coming to a branching point. This mode is intended for presentation, but also for detecting e.g. deadlocks or other kinds of structural conflicts.

- *Interpretation*: In this mode, Expressions used for instance in Assignments and Conditions are evaluated[4] and stored, so that the process will automatically decide how to proceed at a branching point. Still, the user has to provide initial parameters and return values for service calls. This mode is especially useful for testing the various Conditions and Assignments. *(work in progress)*

- *Execution*: This mode integrates with the Rich Service Directory (RSD), meaning that in addition to the *Interpretation* mode services will be invoked using the RSD and their return values will be bound to the respective process properties. Thus the user just has to provide the initial parameters of the process itself. Apart from testing the interworking of the several services in the process, this mode can also be used for actually executing and monitoring the process. *(future work)*

3.11 Import and Merging of Process Diagrams

VSDT diagrams can be imported into and merged with each other. While basically this feature can be used for merging any two or more diagrams, it is most useful for merging diverging versions of the same process diagram, having a common ancestor.

After selecting *Import other VSDT diagrams* from the Import menu, select one or more diagrams to import *from* and one diagram to import *into*. You can also check whether to create a backup of the original target file (recommended), and whether the layout should be imported, too, or only the model data, and whether the algorithm should try to merge identical elements. The latter, of course, only makes sense if the source and target files are different revisions of the same process diagram.

The merging algorithm works by recursively comparing the IDs of the objects to be merged, so these should not be changed in different revisions. Also there are still some issues with conflicting changes, so one should always be sure to create a backup and possibly use a `diff` tool to check whether the changes to the file's XML source look plausible.

3.12 Structure-Based Layout

The Structure Mapping can also be used for calculating the layout of the BPMN diagrams. Compared to the layouting algorithm provided by GMF, this proves especially useful for diagrams containing upstream loops. Still, since the structure-based layout is still in an early stage, the default layouting algorithm still is the one provided by GMF. The structure-based layout can be reached via the Structure View (see Section 1.3).

The recursive layout algorithm is sketched in algorithms 1 and 2. First, the process diagram is transformed to a block structure. Then, the algorithm will step down into

[4]Note that only Expressions given in the VSDT Expression Language can be automatically evaluated, and that by now only simple data types are supported.

the structure and calculate the layout of the "blocks" in a bottom-up way, taking the positions of the nested elements into account. The result of the algorithm is a map holding the bounding box for each element, which can then be used for laying out the elements in the diagram editor.

While the structure-based layout yields good results for well-structured diagrams, i.e. diagrams that can be transformed to a block-structured form, the algorithms is not applicable for non-structured diagram.

Algorithm 1 CreateLayoutMap(*diagram*)

```
structuredDiagram ← Structurize(diagram)
layoutMap ← create empty map
hint ← (0, 0)
for pool in structuredDiagram do
  box ← CalculateBox(pool, hint, layoutMap)
  hint ← (0, box.bottom)
end for
return layoutMap
```

Algorithm 2 CalculateBox(*element*, *topLeft*, *layoutMap*)

```
hint ← topLeft
if element is atomic then
  box ← (topLeft.x, topLeft.y, Width(element), Height(element))
else
  if element is sequence then
    height ← 0
    for child in element do
      box ← CalculateBox(child, hint, layoutMap)
      hint ← (box.left, box.top)
      height ← Max(height, box.height
    end for
    box ← (topLeft.x, topLeft.y, hint.x − topLeft.x, height)
  end if
  // similar for block, loop, subprocess, event handler, etc.
end if
insert (element, box) into layoutMap
return box
```

4 JIAC Node Plugin

Bibliography

[1] Thomas Gschwind, Jana Koehler, and Janette Wong. Applying patterns during business process modeling. In *BPM '08: Proceedings of the 6th International Conference on Business Process Management*, pages 4–19, Berlin, Heidelberg, 2008. Springer-Verlag. 16