

DSKETCH User Guide

Version **20090622**

Date **June 2009**

Author **Florian Brieler**

Email **fbrieler@gmx.de**

Contents

- What is DSKETCH? 3
- Getting Started 3
- The Editor 4
 - The Toolbar 6
 - The Menu 10
 - Selections 13
 - Drag'n Drop 14
- Create Your Own Editors 15
 - Creating a sketch editor step-by-step 15
 - The Designer 15
 - The Designer Toolbar 16
 - The Designer Tree Structure 17
- Cheat Sheets 18
 - Petri nets 18
 - Nassi-Shneiderman diagrams (NSD) 19
 - GUI builder 20
 - Tic-tac-toe 21
 - Boolean logic gates 21
 - State charts 22

What is DSKETCH?

DSKETCH is an approach to the research field of sketching. It allows for the generation of sketch-enabled diagram editors from specifications. The generated editors allow for unrestricted sketching, and recognize and analyze the sketch in order to get the most out of the sketch.

Refer to the publications on DSKETCH in order to find out more about its key characteristics.

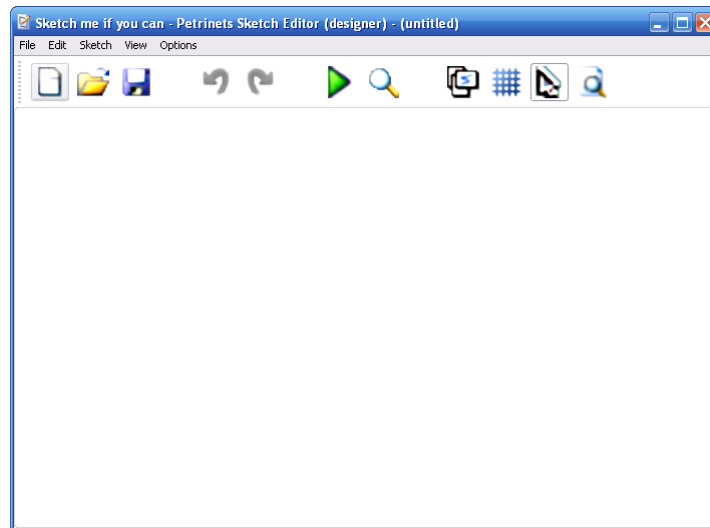
Getting Started

The following list shows all necessary steps in order to get DSKETCH started:

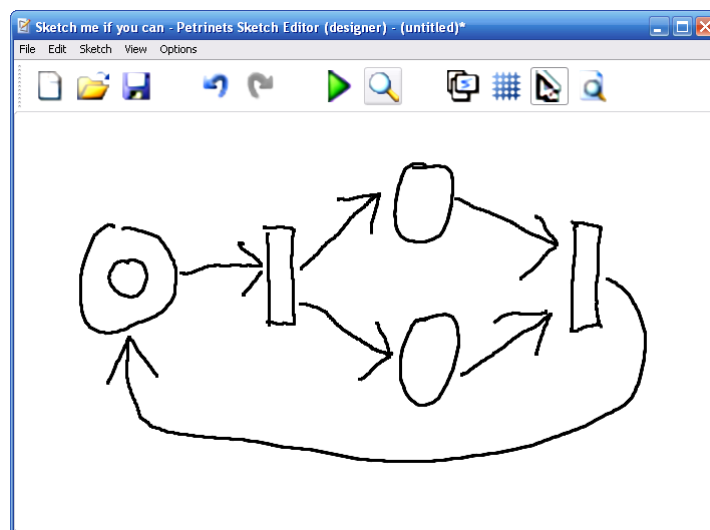
1. Download the latest version of DSKETCH at <http://www.unibw.de/inf2/DiaGen/dsketch> and extract all of its contents into an empty folder on your hard drive.
2. Get Java Standard Edition, version 6 or higher from <http://java.sun.com/javase/>. The latest version is recommended. If you only want to run the examples, the Java Runtime Environment (JRE) is sufficient. If you want to create your own sketch editors, the Java Development Kit (JDK) is required.
3. Make sure that your `[java-installation-folder]/bin` folder is set in your system's path variable. This is required for the included shell scripts to run. Note: if you open your system's command shell and type in `java -version`, this command must execute.
4. That's it – you can now run the examples by executing the shell scripts contained in the folder where you extracted the download package.

The Editor

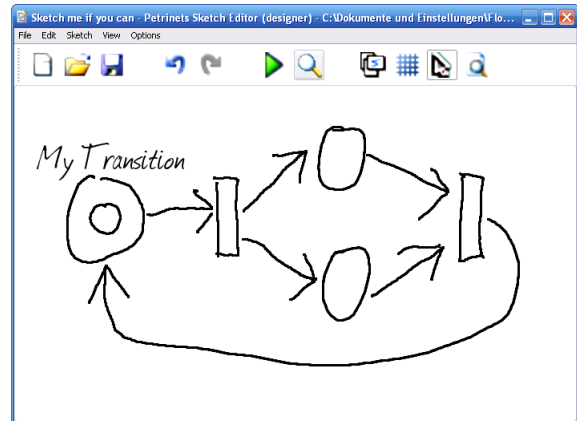
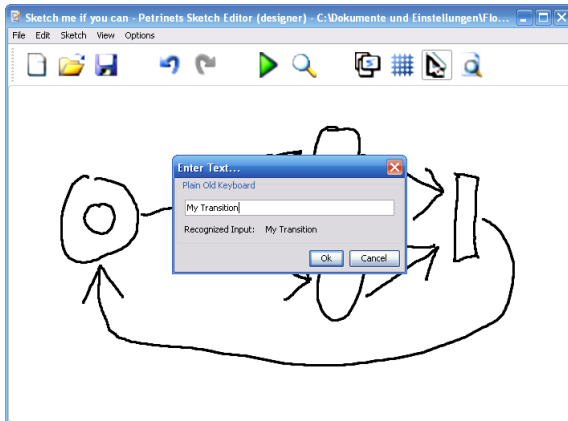
All generated editors have the same GUI and work similarly. If you run an editor for the first time (in this case, the Petri net sketch editor included in the download package) a window is shown that contains an empty canvas to draw on.



You can draw on the canvas according to the specification of your diagram language.



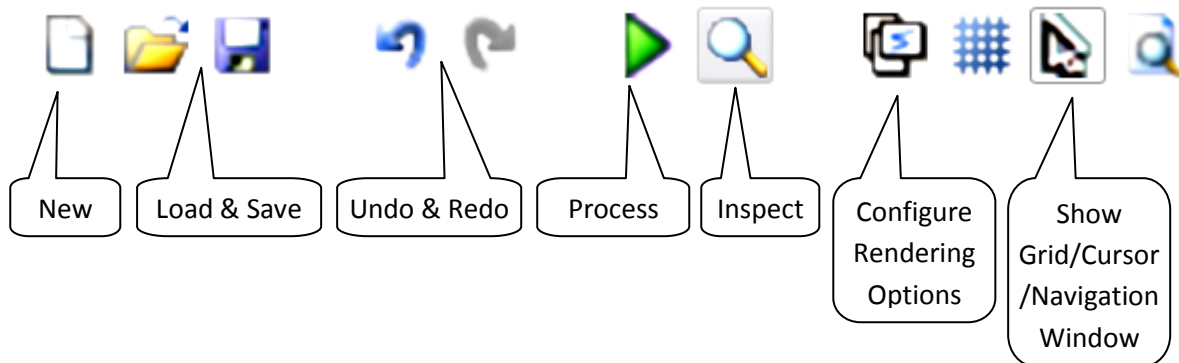
By making a double-tap or double-click with the secondary button of your input device anywhere on the canvas, a popup window appears which allows you to enter text at exactly this position. Note that you cannot directly write on the canvas.



The application is equipped with several text input recognizers, which you can set in the options menu (page 11).

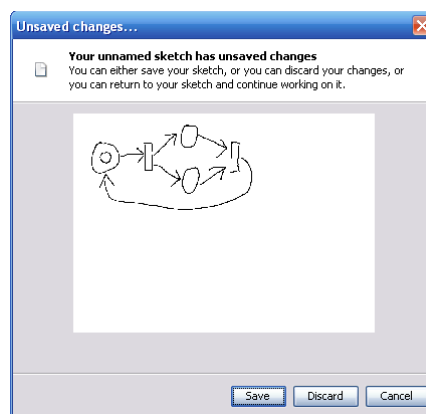
The Toolbar

The toolbar contains the following commands



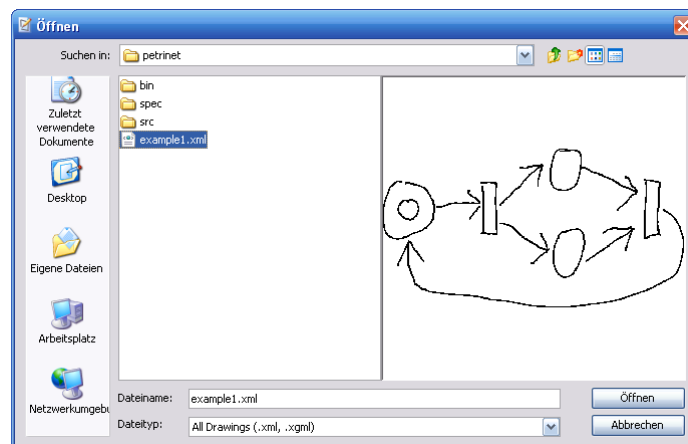
New

Use this button to erase the complete sketch and get an empty canvas to draw on. In case that your current sketch has unsaved changes the following dialog pops up and allows you to save your changes, discard it, or cancel the process.



Load & Save

These two buttons provide you with the standard functionality of saving a sketch or loading it. A little preview window helps you to decide for the right file to load from or save to.



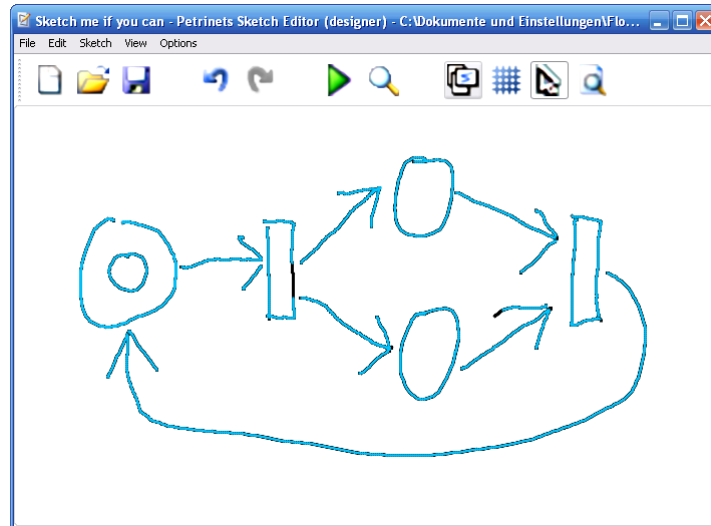
In case that you want to load a file from your hard drive, and your current sketch has unsaved changes, the same dialog pops up as for the New command.

Undo & Redo

These two standard buttons allow you to undo or redo your last strokes and text drawn on the canvas. The granularity is set to one stroke or one text per undo/redo operation.

Process

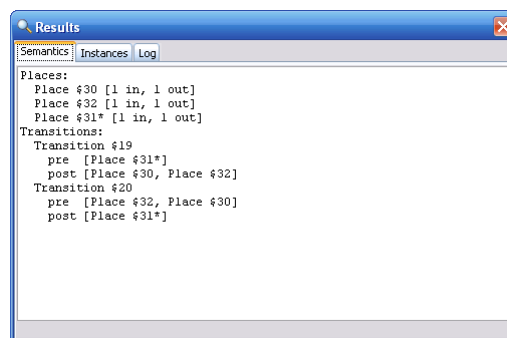
This button starts the recognition and analysis of your sketch. That part of your sketch that could be included in the final result is highlighted.

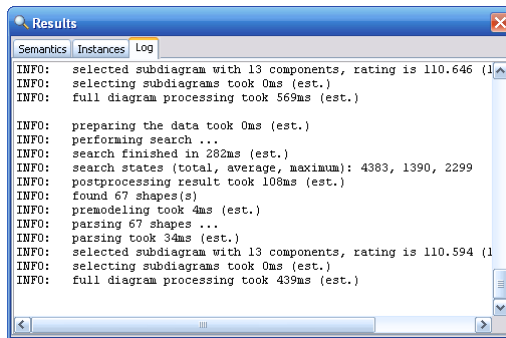
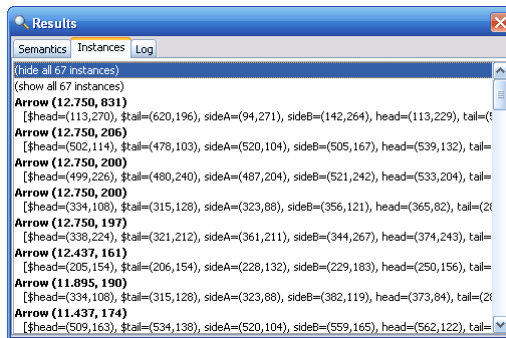


The detailed result of diagram processing can be inspected with the Inspect button

Inspect

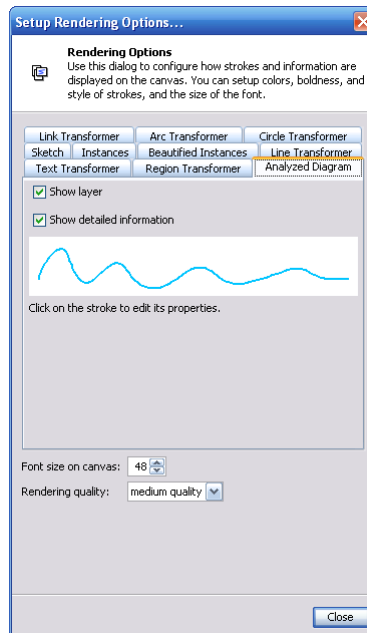
Clicking on this button opens a non-modal popup window that shows you information about the last run of diagram processing. It contains three tabs: Semantics, Instances and Log. The first shows you a textual representation of the semantics represented by that part of the sketch that is included in the final result. The second tab shows you a list of all instances of shapes that have been recognized, sorted by their rating. This list represents the intermediate result between recognition and analysis. You can click on a single instance in order to get it highlighted. Finally, the third tab shows you the complete log since you started the sketch editor.





Configure Rendering Options

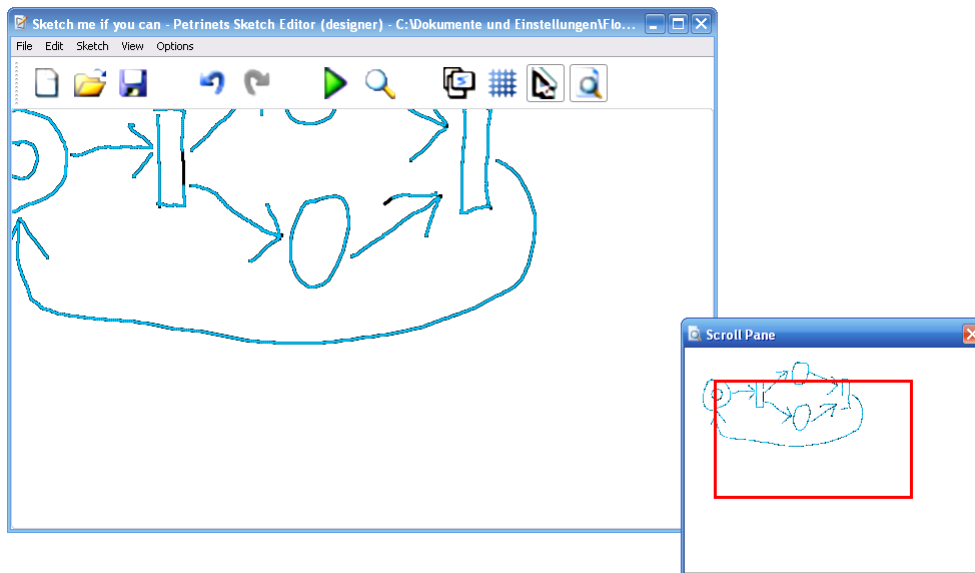
This button brings up a dialog window which allows you to precisely set how your sketch and related information is rendered on the canvas. Each tab looks identical and allows you to configure one individual layer. You can set whether that layer is to be shown, if detailed information is to be shown, and what stroke is used to render that layer. The detailed information depends on the individual layer. By clicking on the stroke, another dialog window appears, where you can set stroke width, style (solid, dashed, dotted) and color.



Furthermore, this dialog allows you to set the font size of text drawn on the canvas and the overall rendering quality.

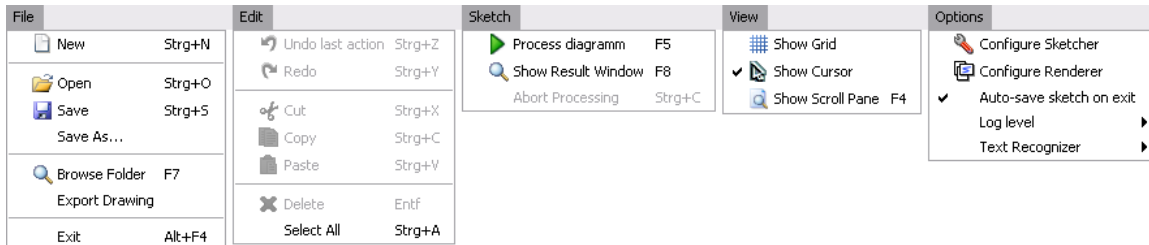
Show Grid/Cursor/Navigation Window

The first of three buttons allows you to show or hide a background grid which can help you in sketching more precisely and aligned to the axes. It is hidden by default. The second button allows you to show or hide a mouse cursor whenever it is on the canvas. The cursor is shown by default; however, you may find it more convenient to hide it when you sketch with a stylus. The third button opens a non-modal popup window that shows your complete sketch. The red rectangle indicates which part of your sketch is currently visible on the canvas. You can drag this rectangle to make another part of your sketch visible.



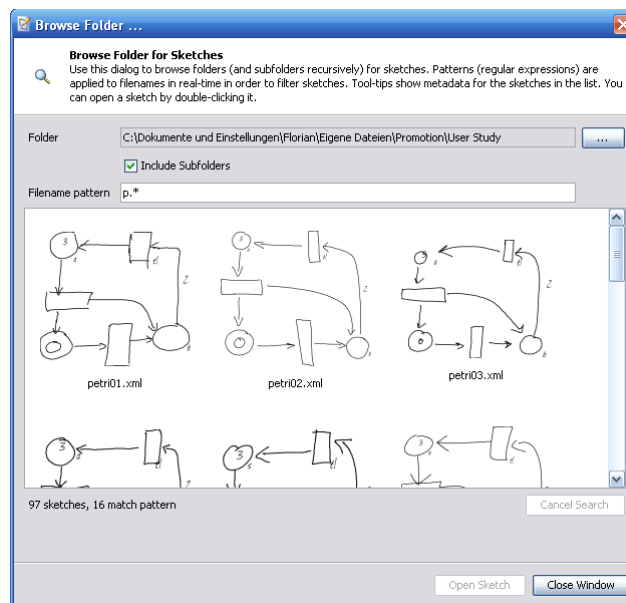
The Menu

The menu contains all the commands of the toolbar, and adds some additional commands, explained in this section. Hotkeys which allow for faster access of commands are shown. Commands not applicable at a given moment are grayed out.

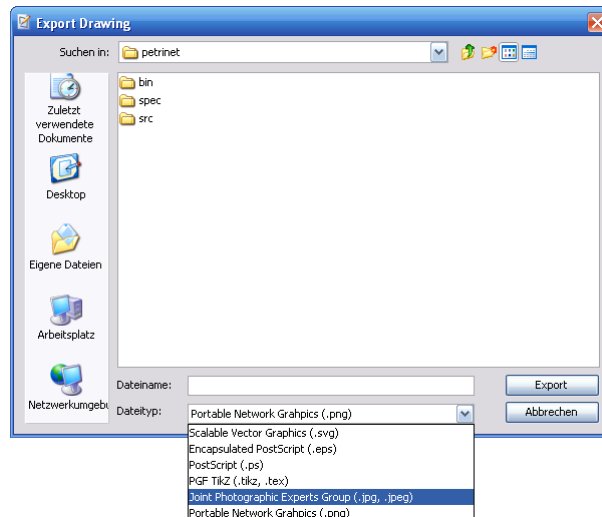


File Menu

This menu contains the following items which are identical to the toolbar: New, Open, and Save. The Browse Folder command opens a new window where all sketches contained in a folder (and its subfolders) are displayed. Individual sketches can be opened. The window is modal. It keeps its state while closed. This way, upon re-opening the window, the same sketches are shown.



The Export drawing command allows exporting the drawing to another than the application's native xml format. Note that not every format is fully supported.



The Exit command exits the application without further prompt.

Edit Menu

This menu contains the following items which are identical to the toolbar: Undo, and Redo. The commands Cut, Copy, Paste, and Delete all relate to a selection (page 13) and work as in virtually any other application. Finally, the Select all command selects all contents on the canvas.

Sketch Menu

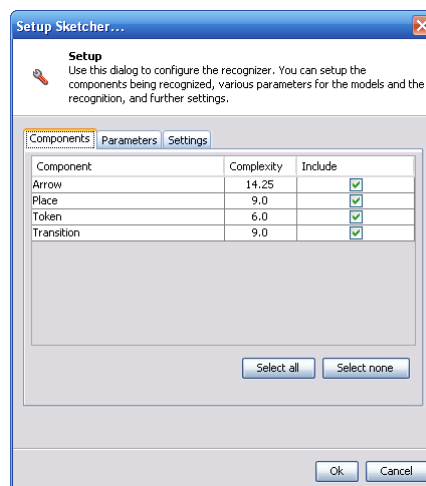
This menu contains the following items which are identical to the toolbar: Process, and Inspect. The third command allows for aborting of diagram processing. This command is only available while a sketch is processed; otherwise, it is grayed out.

View Menu

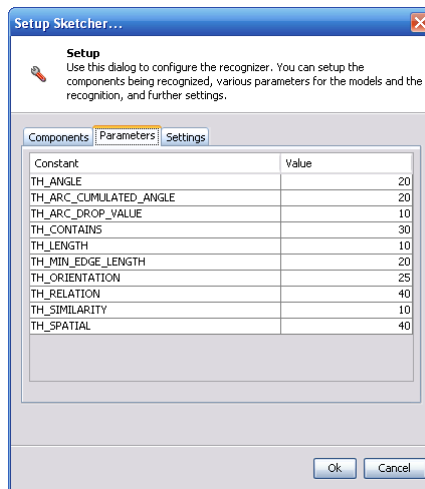
This menu contains the following items which are identical to the toolbar: Show grid, Show cursor, and Show navigation window.

Options Menu

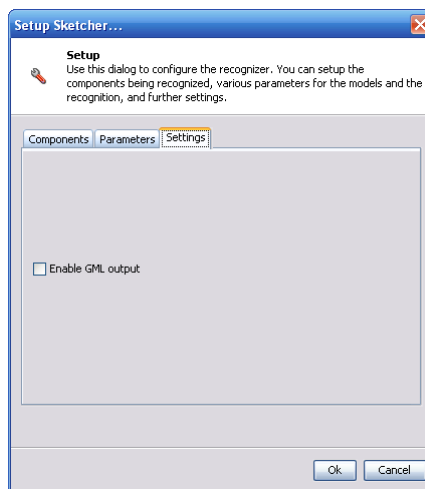
This menu contains the following item which is identical to the toolbar: Configure rendering options. The Configure sketcher command brings up a dialog window where you can set various properties of the sketch processing engine. The first tab shows a list of all shapes specified for your diagram language, and their complexity. You can select and deselect shapes in order to include them in or exclude them from recognition. By default, all shapes are included.



The second tab allows setting the values for the various thresholds included in the recognition process.



The third tab allows enabling and disabling gml output of internal models of the analysis stage (hypergraph model, reduced hypergraph model, and derivation dags).



The third item in the options menu allows setting whether your sketch should be saved automatically when you close the application. This option is enabled by default. The sketch is saved to a file named autosave.xml. The Log level submenu allows to the granularity of the log level for the log shown on the console and in the third tab of the Inspect popup window. The Text recognizer submenu allows to set the text recognizer used to enter text. Recall that this window opens when you make a double-tab or double-click the secondary button of your input device on the canvas (page 4).

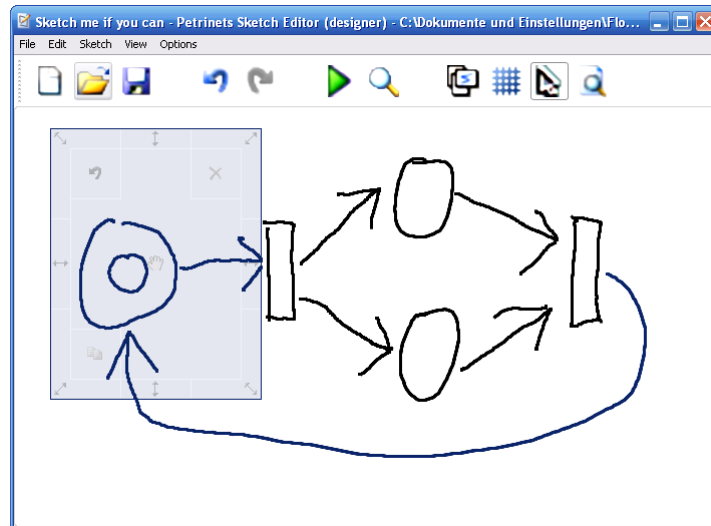
Note: if you are running Microsoft Windows XP Tablet PC Edition on your computer, you can add the operating system's text recognizer in your application. Therefore, close your application and locate the settings.xml file where all your settings are saved. Locate the line that says

```
<setting name="windowsxptablet" value="false" />
```

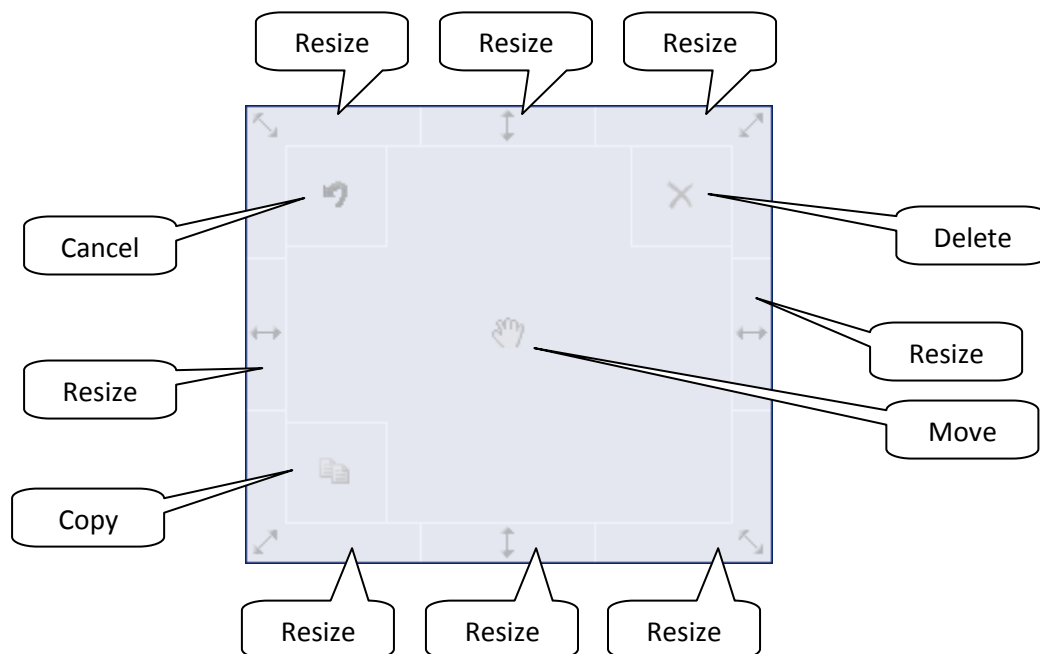
and change the value from "false" to "true". Then restart your application.

Selections

Besides drawing on the canvas and adding text, there is a further way to edit your sketch: selections. You can select a portion of your sketch (including text) by pressing and holding the secondary button of your input device and dragging it over the canvas.



Note that all strokes and text are included in your selection if at least part of them is inside the selection rectangle. In the figure above, the long arrow shaft is included in the selection, for example. If there is a selection, all commands from the edit menu become available (Cut, Copy, Paste, and Delete, page 11). Furthermore, you can do the following things with your selection:

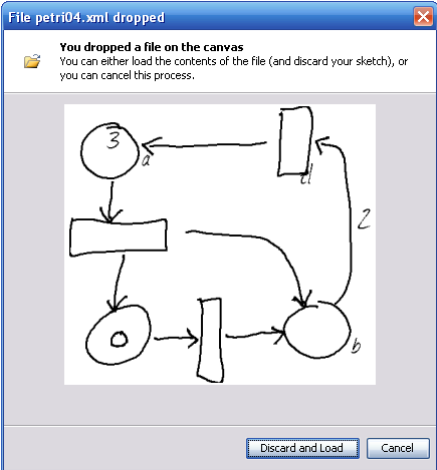


Each command is available inside the full polygon where its icon is placed in, and affects all selected content (stroke and text). The only exceptions are the resize commands: only strokes are resized, text is not. Note that the commands Move, Copy, and Resize require to click and hold the secondary button of the input device, and drag it afterwards, while Delete and Cancel require you to make a

click or tab with the secondary button. Delete deletes all selected content, while Cancel performs an undo on every move and resize operation you have done with your current selection. Also note that the icons inside the selection are hidden automatically if there is not enough space to show them. However, the functionality is unchanged.

Drag'n Drop

The editors allow for drag'n drop of files right on the canvas. If the dropped file contains a valid sketch, a preview window is shown which offers two options: Either to discard the current sketch and the load the dropped file; or to cancel the process.



If the dropped file does not contain a valid sketch, a respective message is shown.

Create Your Own Editors

All examples included in the download package have the same folder structure, which is not mandatory, but recommended. For each example there are three subfolders: bin, spec, and src. The spec folder contains the file that specifies the complete sketch editor, recognizer and analysis. This file has always the extension .dsketch. The necessary source code for a sketch editor is generated from this specification file only. The code resides in the src subfolder. For each of the included examples there is one additional source file, Semantics.java, which is hand-coded and contains the code to compute the semantics of a sketched diagram. All generated files and Semantics.java are compiled to the bin subfolder. The main method of each sketch editor is always located in class SketchEditor. Running a sketch editor requires lib/framework.jar and lib/ext/jdom-1.0.jar in the classpath. If you want to use the Microsoft handwriting recognizer (page 11), you furthermore need reco.dll and lib/dll/cnct.dll to be accessible. For your convenience, jar files have been created for all six examples where the classpath information is already included.

Creating a sketch editor step-by-step

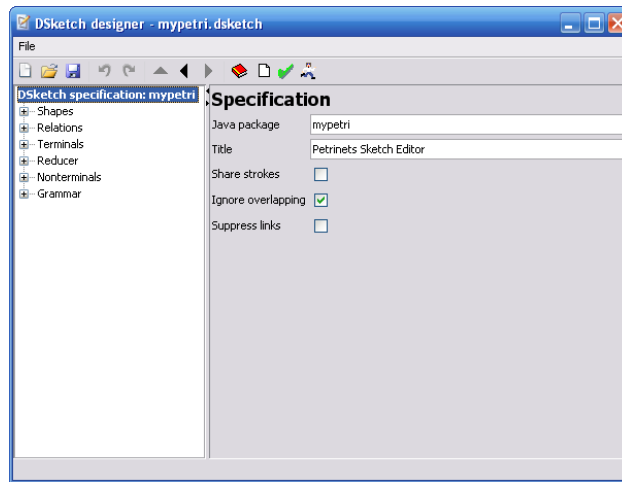
To create a new sketch editor, the following steps are required. Note that we assume that you stick to the folder structure described above:

1. Create a new subfolder in your DSKETCH extraction folder.
2. Create subfolders bin, spec, and src within your new folder from step 1.
3. Create the specification file by using the designer, and create code from it (see below).
4. Create the file Semantics.java in a subfolder of src matching your package structure and add all necessary code to compute the semantics.
5. Compile all source files in src to bin.
6. Optional: if you like, you can assemble a jar file and/or write a respective shell script to start your editor.

As the examples show, there is the -o[folder] command-line argument passed to SketchEditor.class, where [folder] represents the subfolder of the diagram language. Using this option, the working directory of the sketch editor is set. This working directory is the default for load and save, and will contain gml files generated by sketch processing, the configuration file settings.xml and autosave.xml, if applicable. Using different working directories it is even possible that different users with different settings use the same editor.

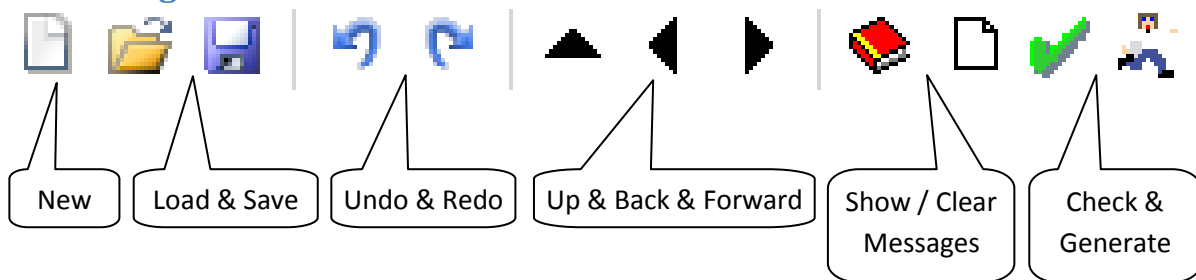
The Designer

The Designer is used to create the .dsketch specification files, edit these files and create code from it. Its jar file resides in the lib subfolder of your DSKETCH extraction folder as lib/designer.jar. Shell scripts are present for your convenience, however, the jar file can also be directly started. When started, and a specification file is loaded, it looks like the following.



The complete specification is accessible by the tree structure on the left-hand side of the window. The right-hand side of the window with the gray background is context-sensitive, and allows modifying the contents of the currently selected tree node.

The Designer Toolbar



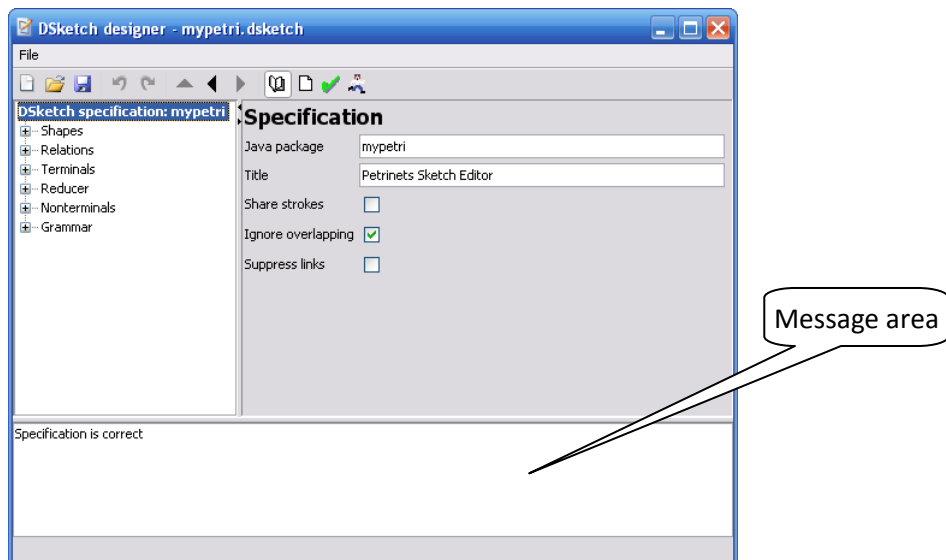
The first five buttons work like their counter-parts in the DSKETCH editors. Undo, and Redo only refer to changes made to the tree structure, and do not consider changes made to individual nodes (right-hand side of the window).

Up & Back & Forward

These three buttons allow navigating through the tree structure. Up navigates to the parent node of the current node, if there is one. Back and Forward do not navigate to preceding or succeeding nodes, but, similar to the functionality known from web browsers, backwards and forwards through the history of visited nodes.

Show Messages & Clear Messages

When checking the document or generating output (see below), a message area pops up that gives information about the process. With Show Messages this area can be shown and hidden manually. With Clear Messages the contents of the area are cleared.



Check Document & Generate

Both buttons check the document, i.e., check whether the specification is syntactically correct. Additionally, Generate generates the desired source code. By default, the code is generated to the folder in which the specification file resides. Using the `-o [folder]` command line argument, the output folder can be changed to the specified `[folder]`.

The Designer Tree Structure

As mentioned before, the left-hand side of the designer window comprises a tree structure that represents the specification. The root node, always without icon and labeled with **DSkech specification: ...** has six children, reflecting the six main parts of the spec:

- Shapes – the shapes the visual language consists of
- Relations – defined between the shapes (or, to be more precise, between the attachment areas of the shapes)
- Terminals – produced by the reducer
- Reducer – the reduction rules applied by the reducer in order to produce terminals from the recognized shapes and relations
- Nonterminals – used by the parser
- Grammar – the hypergraph grammar defining the syntax of the language

While these six nodes always exist, their children depend on the actual diagram language. Clicking with the right mouse button on tree nodes pops up a small menu that allows for Cut, Copy, and Paste functionality. For example, if you have a shape called *Place* and you want to create a similar shape, right-click on *Place*, select Copy, right-click on its parent node *Shapes*, and select Paste.

Now open the included specification files in order to get an overview of how specifications look like, and consider the publications on DSKETCH in order to find out about the meaning of the different specification aspects.

Cheat Sheets

The following pages contain information about the syntax of the six examples included in the download package, by means of an example and a small reference.

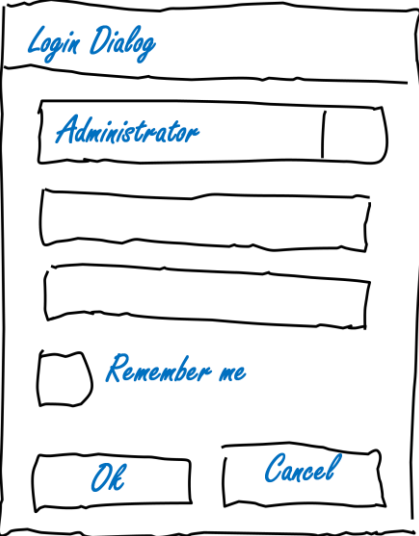
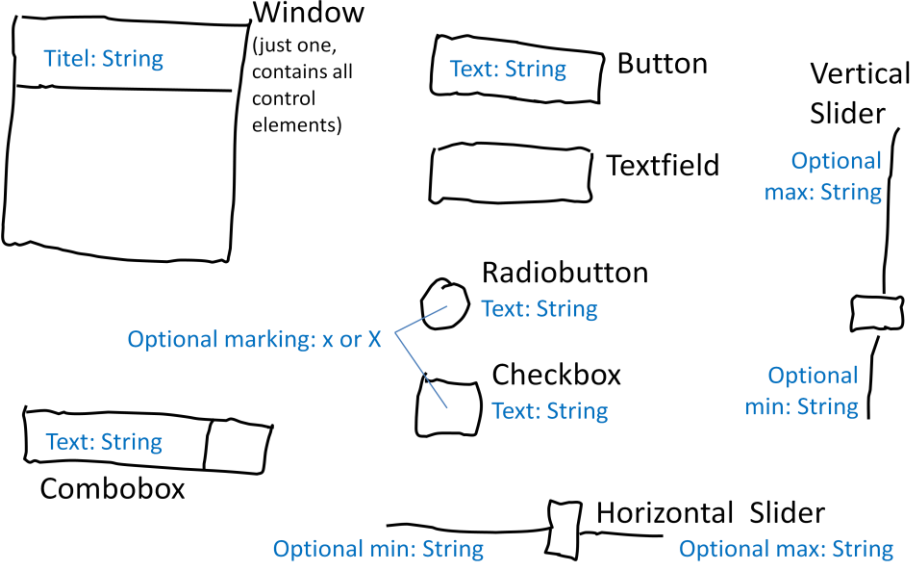
Petri nets

<p>Example</p>	
<p>Reference</p>	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Place</p> <p>Optional capacity: Positive</p> <p>Optional name: String</p> </div> <div style="text-align: center;"> <p>Arrow</p> <p>(only between place- transition or transition- place)</p> <p>Optional cost: Positive</p> </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="text-align: center;"> <p>Token</p> <p>(only in places)</p> </div> <div style="text-align: center;"> <p>Transition</p> <p>Optional name: String</p> </div> </div>

Nassi-Shneiderman diagrams (NSD)

<p>Example</p>	
<p>Reference</p>	

GUI builder

<p>Example</p>	
<p>Reference</p>	 <p>Window (just one, contains all control elements)</p> <p>Titel: String</p> <p>Text: String Button</p> <p>Textfield</p> <p>Radiobutton Text: String</p> <p>Checkbox Text: String</p> <p>Optional marking: x or X</p> <p>Text: String Combobox</p> <p>Vertical Slider Optional max: String</p> <p>Optional min: String</p> <p>Horizontal Slider Optional min: String Optional max: String</p>

Tic-tac-toe

Example	
Reference	

Boolean logic gates

Example	
Reference	<p>Optional nameA: String</p> <p>Gate (binary for & and >=1, unary for 1)</p> <p>Wire (connects output(always right) to input (always left). Names must be set if not connected to gate or inverter)</p> <p>Optional nameB: String</p> <p>Inverter (alternative)</p> <p>Bubble (inverts output, must be present for Inverter or 1, optional for & and >=1)</p>

State charts

<p>Example</p>	<pre> graph TD Start(()) --> ready[ready] ready --> computing[computing] computing -- not finished --> ready computing -- done --> End((())) </pre>
<p>Reference</p>	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Initial state (no incoming arrows, exactly one outgoing arrow)</p> <p>Final state (no outgoing arrows)</p> </div> <div style="width: 45%;"> <p>State</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-bottom: 10px;">Name: String</div> <p>Arrow</p> </div> </div>