

Asciidoctor Diagram

Supported Diagram Types

brought to you with ♥ by barthel

version: df6bb209

This document describes and shows all diagram types provided by AsciiDoctor Diagram.

Table of Content

| | |
|---|----|
| 1. Introduction and goals | 5 |
| 2. ASCIIToSVG | 6 |
| 2.1. Internal diagram source | 6 |
| 2.2. External diagram source file | 6 |
| 3. Barcodes | 7 |
| 3.1. bookland (ISBN) | 7 |
| 3.2. codabar | 7 |
| 3.3. code25 | 7 |
| 3.4. code25iata | 7 |
| 3.5. code25interleaved | 7 |
| 3.6. code39 | 8 |
| 3.7. code93 | 8 |
| 3.8. code128 | 8 |
| 3.9. code128a | 8 |
| 3.10. code128b | 8 |
| 3.11. code128c | 9 |
| 3.12. ean8 | 9 |
| 3.13. ean13 | 9 |
| 3.14. gs1_128 | 9 |
| 3.15. qrcode | 9 |
| 3.16. upca | 10 |
| 4. Blockdiag | 11 |
| 4.1. actdiag | 11 |
| 4.2. blockdiag | 13 |
| 4.3. nwdiag | 14 |
| 4.4. seqdiag | 18 |
| 5. BPMN | 19 |
| 5.1. Internal diagram source | 19 |
| 5.2. External diagram source file | 19 |
| 6. Bytefield | 20 |
| 6.1. Internal diagram source | 20 |
| 6.2. External diagram source file | 20 |
| 7. Diagrams as (Python) Code | 21 |
| 7.1. Internal diagram source | 21 |
| 7.2. External diagram source file | 21 |
| 8. Ditaa | 23 |
| 8.1. Internal diagram source | 23 |
| 8.2. External diagram source file | 23 |

| | |
|--|----|
| 9. Dpic | 24 |
| 9.1. Internal diagram source | 24 |
| 9.2. External diagram source file | 24 |
| 10. ERD | 25 |
| 10.1. Internal diagram source | 25 |
| 10.2. External diagram source file | 25 |
| 11. Gnuplot | 26 |
| 11.1. Internal diagram source | 26 |
| 11.2. External diagram source file | 26 |
| 12. graphviz | 28 |
| 12.1. Internal diagram source | 28 |
| 12.2. External diagram source file | 28 |
| 13. meme | 30 |
| 14. mermaid | 31 |
| 14.1. Internal diagram source | 31 |
| 14.2. External diagram source file | 33 |
| 15. mscgen | 34 |
| 15.1. Internal diagram source | 34 |
| 15.2. External diagram source file | 34 |
| 16. Nomnoml | 35 |
| 16.1. Internal diagram source | 35 |
| 16.2. External diagram source file | 35 |
| 17. Pikchr | 37 |
| 17.1. Internal diagram source | 37 |
| 17.2. External diagram source file | 37 |
| 18. PlantUML | 39 |
| 18.1. PlantUML | 39 |
| 18.2. Salt | 40 |
| 19. state-machine-cat (smcat) | 42 |
| 19.1. Internal diagram source | 42 |
| 19.2. External diagram source file | 42 |
| 20. Svgbob | 44 |
| 20.1. Internal diagram source | 44 |
| 20.2. External diagram source file | 44 |
| 21. Symbolator | 46 |
| 21.1. Internal diagram source | 46 |
| 21.2. External diagram source file | 46 |
| 22. Syntrax | 47 |
| 22.1. Internal diagram source | 47 |
| 22.2. External diagram source file | 47 |
| 23. Tikz | 48 |

| | |
|--|----|
| 23.1. Internal diagram source | 48 |
| 23.2. External diagram source file | 49 |
| 24. UMLet | 51 |
| 24.1. Internal diagram source | 51 |
| 24.2. External diagram source file | 51 |
| 25. Vega Lite | 52 |
| 25.1. Internal diagram source | 52 |
| 25.2. External diagram source file | 52 |
| 26. Vega | 53 |
| 26.1. Internal diagram source | 53 |
| 26.2. External diagram source file | 53 |
| 27. WaveDrom | 55 |
| 27.1. Internal diagram source | 55 |
| 27.2. External diagram source file | 55 |
| Bibliography | 56 |

Chapter 1. Introduction and goals

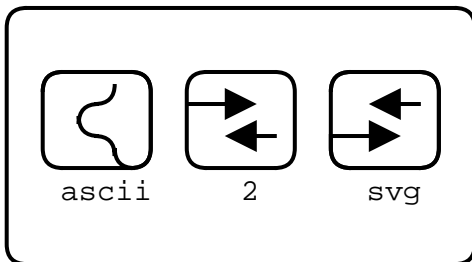
This document should give an overview over all supported diagram types provided by AsciiDoctor Diagram^[DIAG].

Chapter 2. ASCIIToSVG

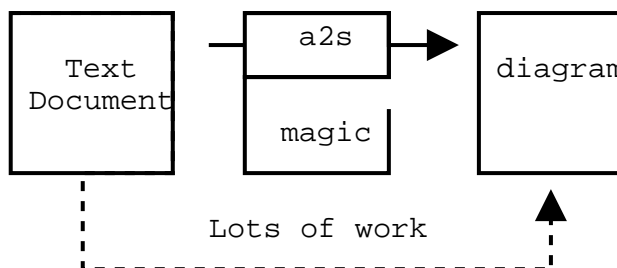
ASCIIToSVG parses ASCII art diagrams, attempting to convert them to an aesthetically pleasing SVG output.

— ASCIIToSVG, <https://github.com/asciitosvg/asciitosvg>

2.1. Internal diagram source



2.2. External diagram source file



Chapter 3. Barcodes

The barcode extension provides barcode rendering. Barcode macros can be specified using blocks, inline macros or block macros.

— AsciiDoctor Diagrams, <https://docs.asciidoctor.org/diagram-extension/latest/#barcode>

3.1. bookland (ISBN)



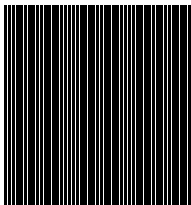
3.2. codabar



3.3. code25



3.4. code25iata



3.5. code25interleaved



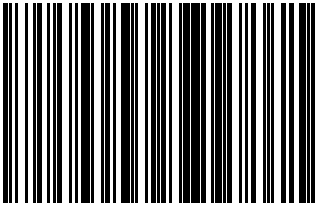
3.6. code39



3.7. code93



3.8. code128



3.9. code128a



3.10. code128b



3.11. code128c



3.12. ean8



3.13. ean13



3.14. gs1_128

No valid data because of <FNC1>.

3.15. qrcode



3.16. upca



Chapter 4. Blockdiag

blockdiag and its family generate diagram images from simple text files.

— Takeshi KOMIYA, <http://blockdiag.com/en/index.html>

blockdiag supports many types of diagrams like

- activity diagram (w/ [actdiag](#)) and
- block diagram (w/ [blockdiag](#)),
- logical network diagram (w/ [nwdiag](#)).
- sequence diagram (w/ [seqdiag](#)),

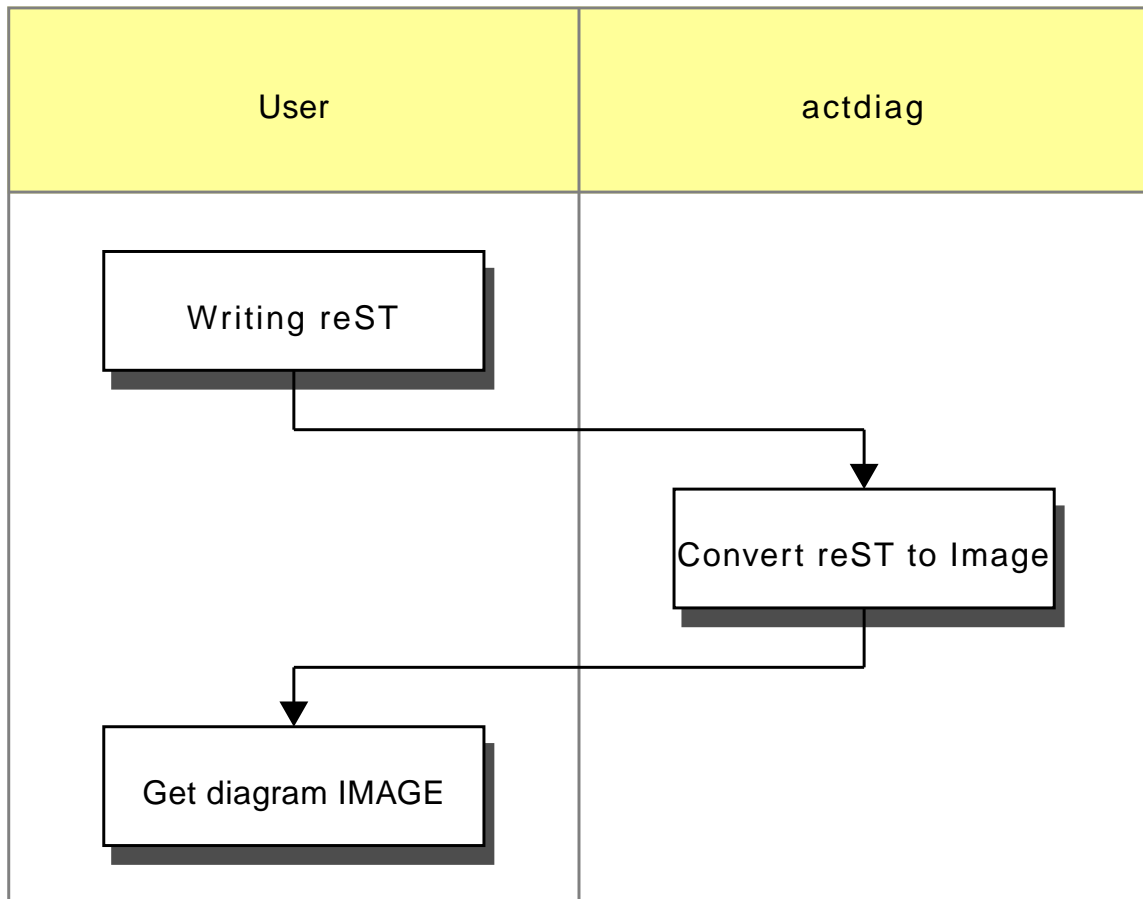
All these tools layouts diagram elements automatically and generates beautiful diagram images from simple text format (similar to graphviz's DOT format).

4.1. actdiag

actdiag is a simple activity-diagram image generator and generates activity-diagram images from .diag files (similar to graphviz's DOT files).

— Takeshi KOMIYA, <http://blockdiag.com/en/actdiag/index.html>

4.1.1. Internal diagram source



4.1.2. External diagram source file



4.2. blockdiag

blockdiag generates block-diagram images from .diag files (similar to graphviz's DOT files).

— Takeshi KOMIYA, <http://blockdiag.com/en/blockdiag/index.html>

4.2.1. Internal diagram source



4.2.2. External diagram source file



4.3. nwdiag

nwdiag generates network-diagram images from .diag files (similar to graphviz's DOT files).

— Takeshi KOMIYA, <http://blockdiag.com/en/nwdiag/index.html>

4.3.1. Internal diagram source



And, `nwdiag` package includes more scripts called `rackdiag` and `packetdiag`.

4.3.2. rackdiag

`rackdiag` generates rack-structure diagram images:

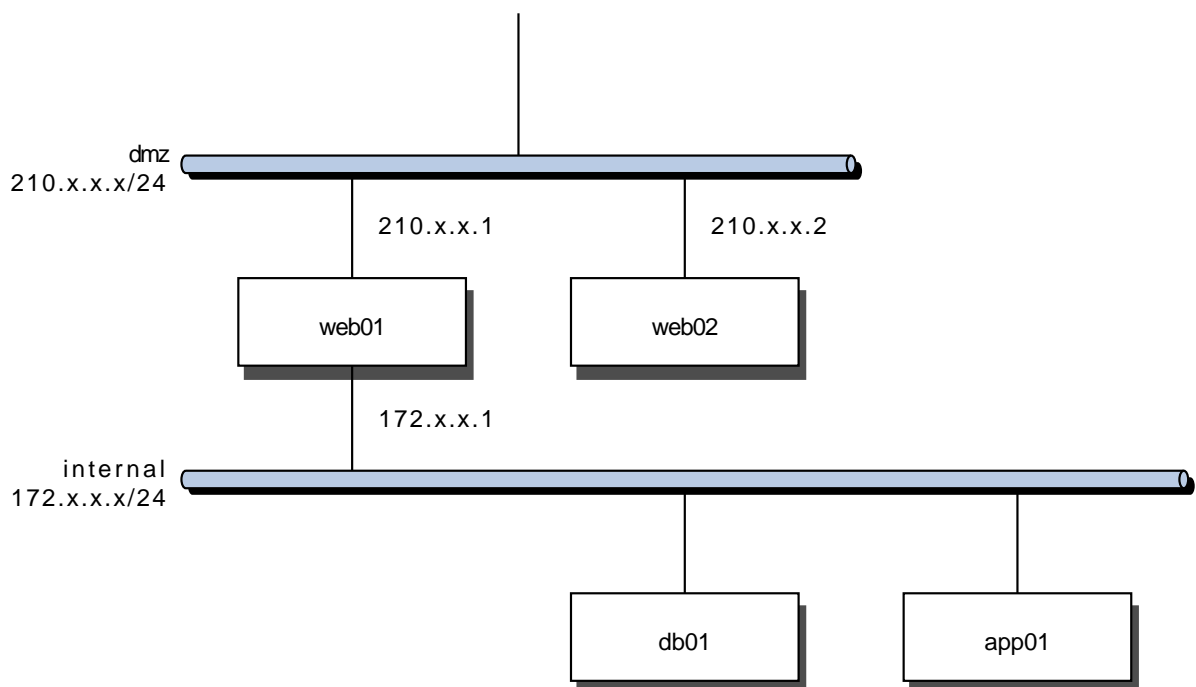


4.3.3. packetdiag

packetdiag generates packet header diagram images:



4.3.4. External diagram source file



4.4. seqdiag

seqdiag generates sequence-diagram images from .diag files (similar to graphviz's DOT files).

— Takeshi KOMIYA, <http://blockdiag.com/en/seqdiag/index.html>

4.4.1. Internal diagram source



4.4.2. External diagram source file



Chapter 5. BPMN

BPMN everywhere, for everyone

Create, embed and extend BPMN diagrams.

— bpmn.io, <https://bpmn.io/toolkit/bpmn-js/>

5.1. Internal diagram source



5.2. External diagram source file

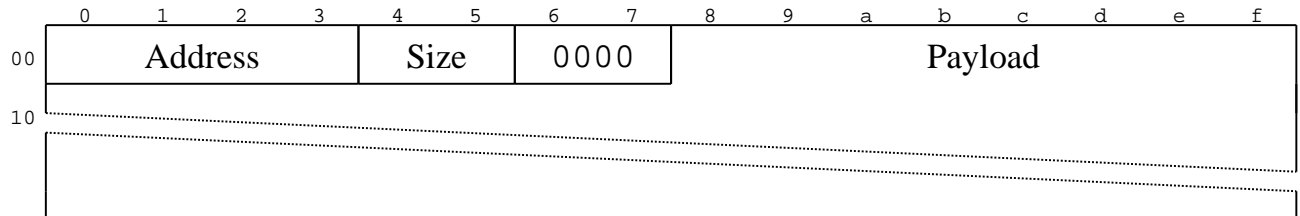


Chapter 6. Bytefield

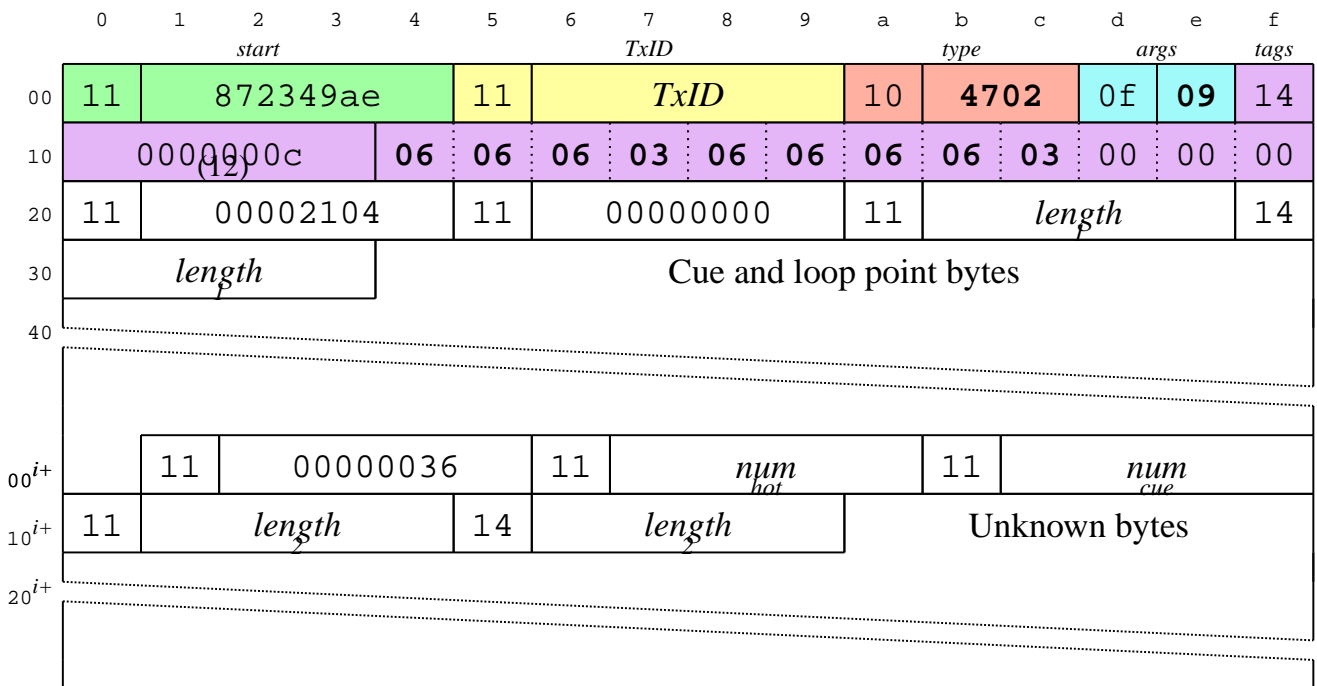
Generating byte field diagrams.

— bytefield, <https://github.com/Deep-Symmetry/bytefield-svg>

6.1. Internal diagram source



6.2. External diagram source file



Chapter 7. Diagrams as (Python) Code

Diagrams — Diagram as Code

Diagrams lets you draw the cloud system architecture in Python code.

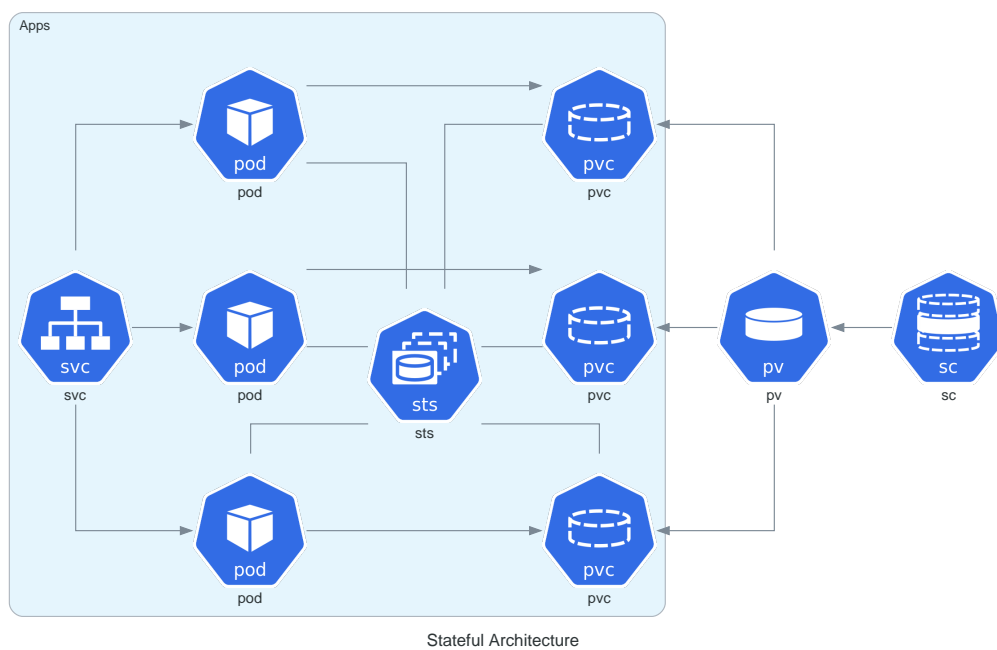
It was born for prototyping a new system architecture without any design tools. You can also describe or visualize the existing system architecture as well.

— Diagrams, <https://diagrams.mingrammer.com/>

7.1. Internal diagram source



7.2. External diagram source file



Chapter 8. Ditaa

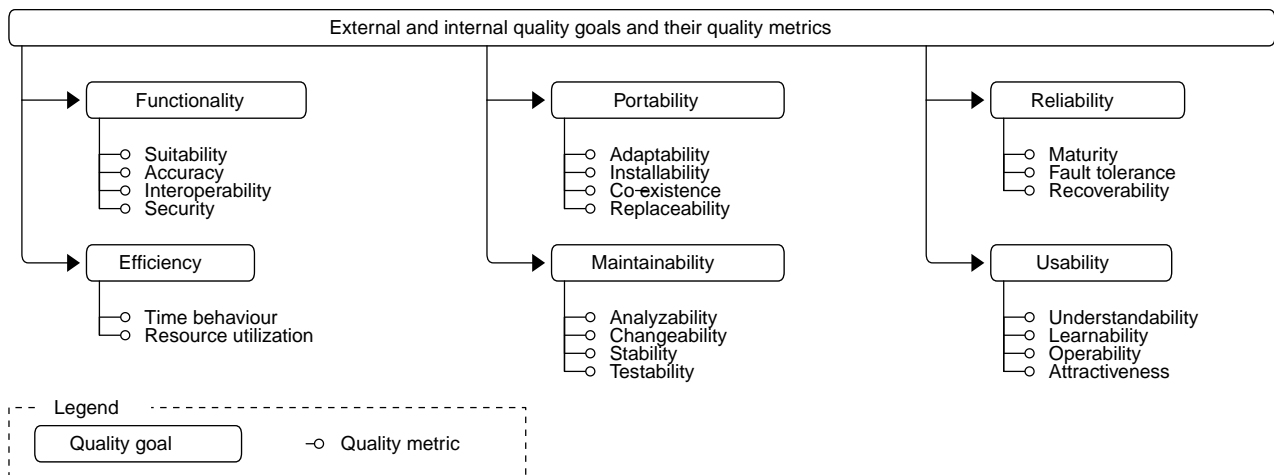
Ditaa is a small command-line utility written in Java, that can convert diagrams drawn using ascii art into proper bitmap graphics.

— ditaa, <http://ditaa.sourceforge.net/>

8.1. Internal diagram source



8.2. External diagram source file

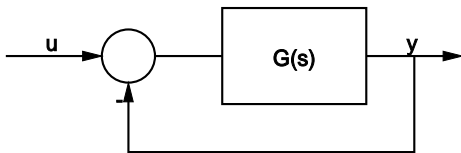


Chapter 9. Dpic

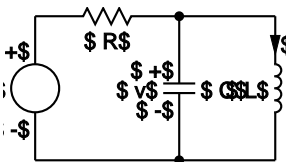
Dpic is an implementation of the pic "little language" for creating line drawings and illustrations for documents, web pages, and other uses.

— J. D. Aplevich, <https://gitlab.com/aplevich/dpic>

9.1. Internal diagram source



9.2. External diagram source file



Chapter 10. ERD

Translates a plain text description of a relational database schema to a graphical entity-relationship diagram.

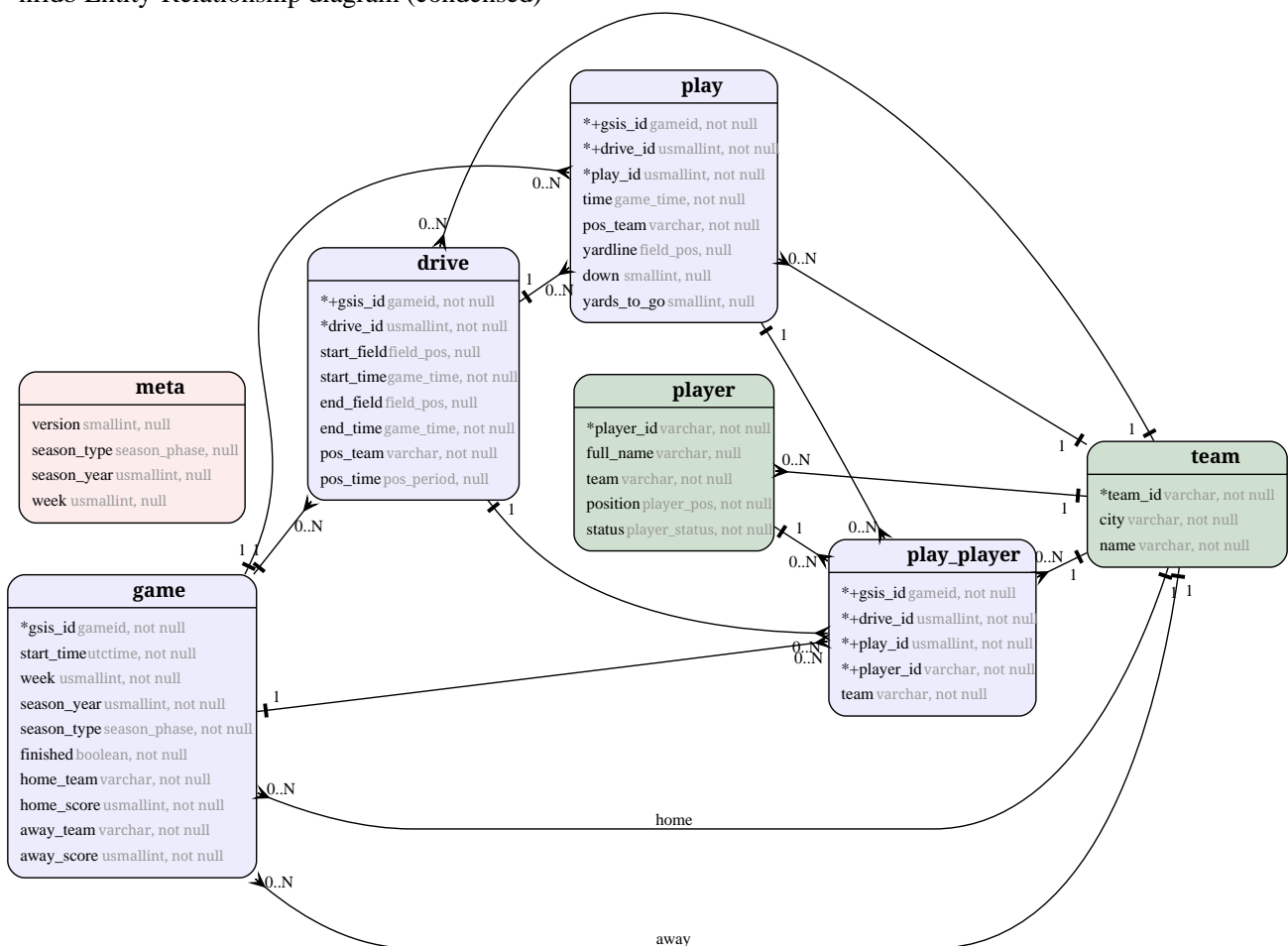
— erd, <https://github.com/kaishuu0123/erd-go>

10.1. Internal diagram source



10.2. External diagram source file

nfldb Entity-Relationship diagram (condensed)



Chapter 11. Gnuplot

Gnuplot is a portable command-line driven graphing utility originally created to allow scientists and students to visualize mathematical functions and data interactively, but has grown to support many non-interactive uses such as web scripting.

— Gnuplot, <http://gnuplot.info/>

11.1. Internal diagram source

Simple Plots



11.2. External diagram source file

Mandelbrot function

`mand({0,0},compl(x,y),30)` —



Chapter 12. graphviz

Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.

— graphviz, <https://graphviz.gitlab.io/>

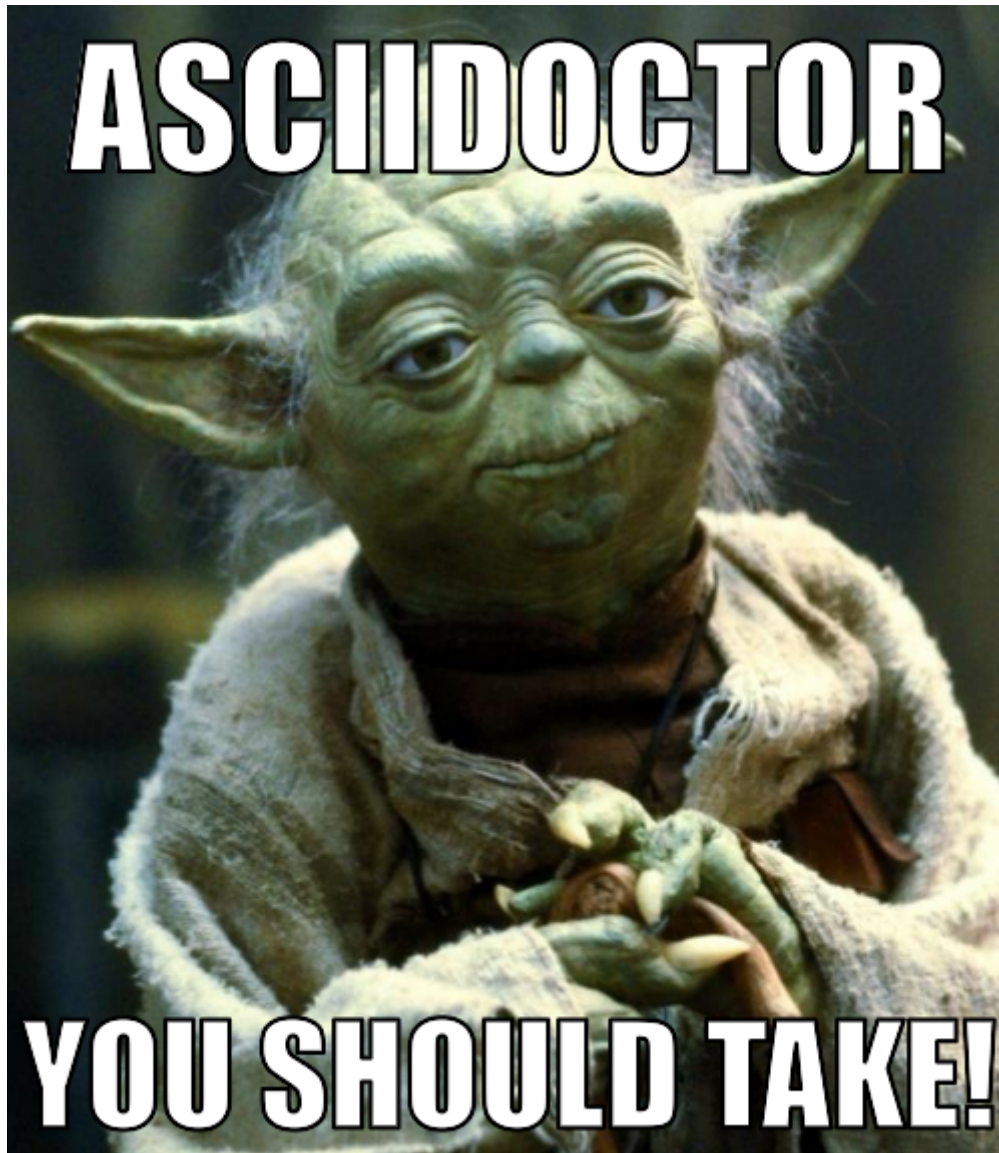
12.1. Internal diagram source



12.2. External diagram source file

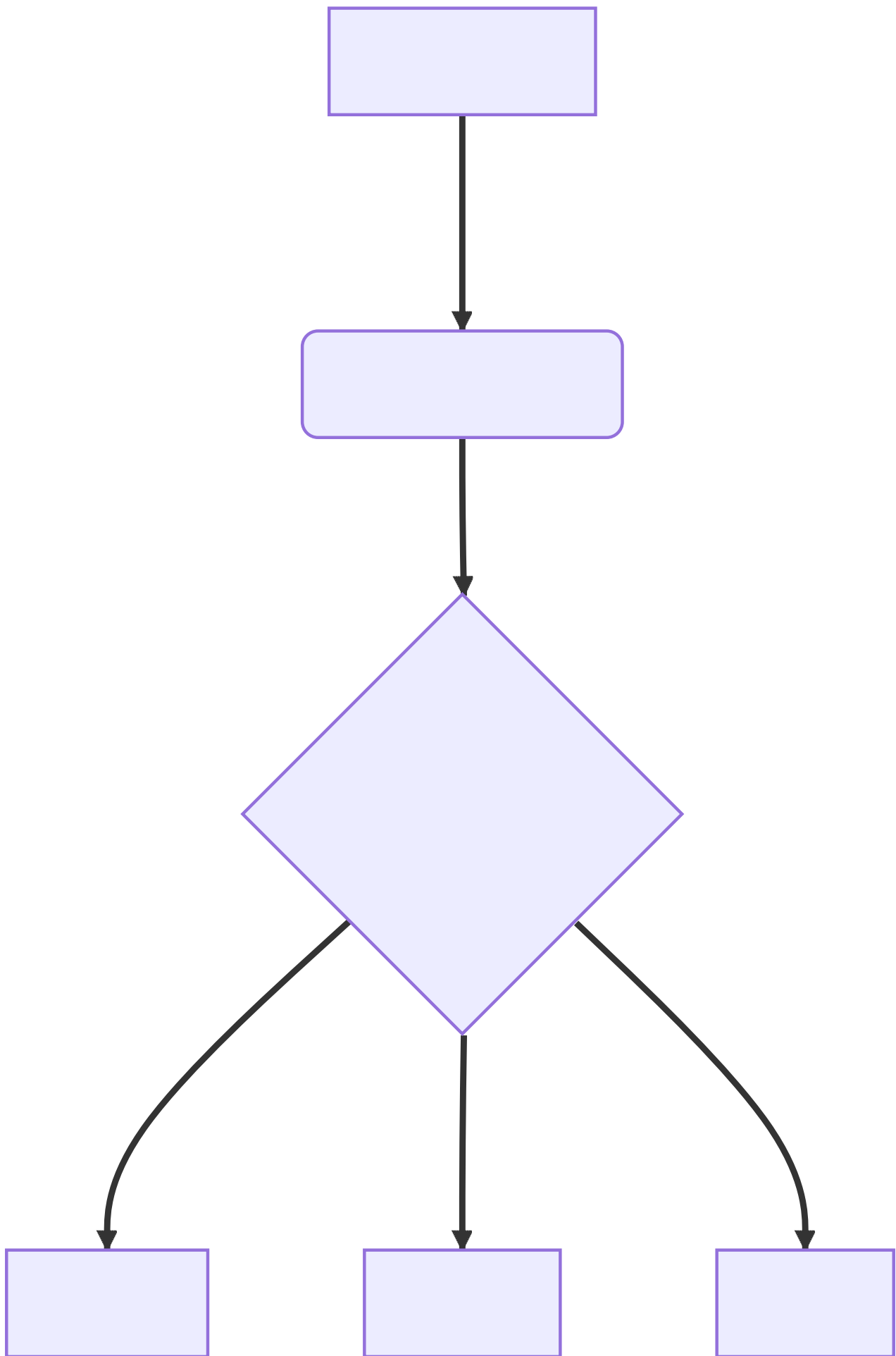


Chapter 13. meme

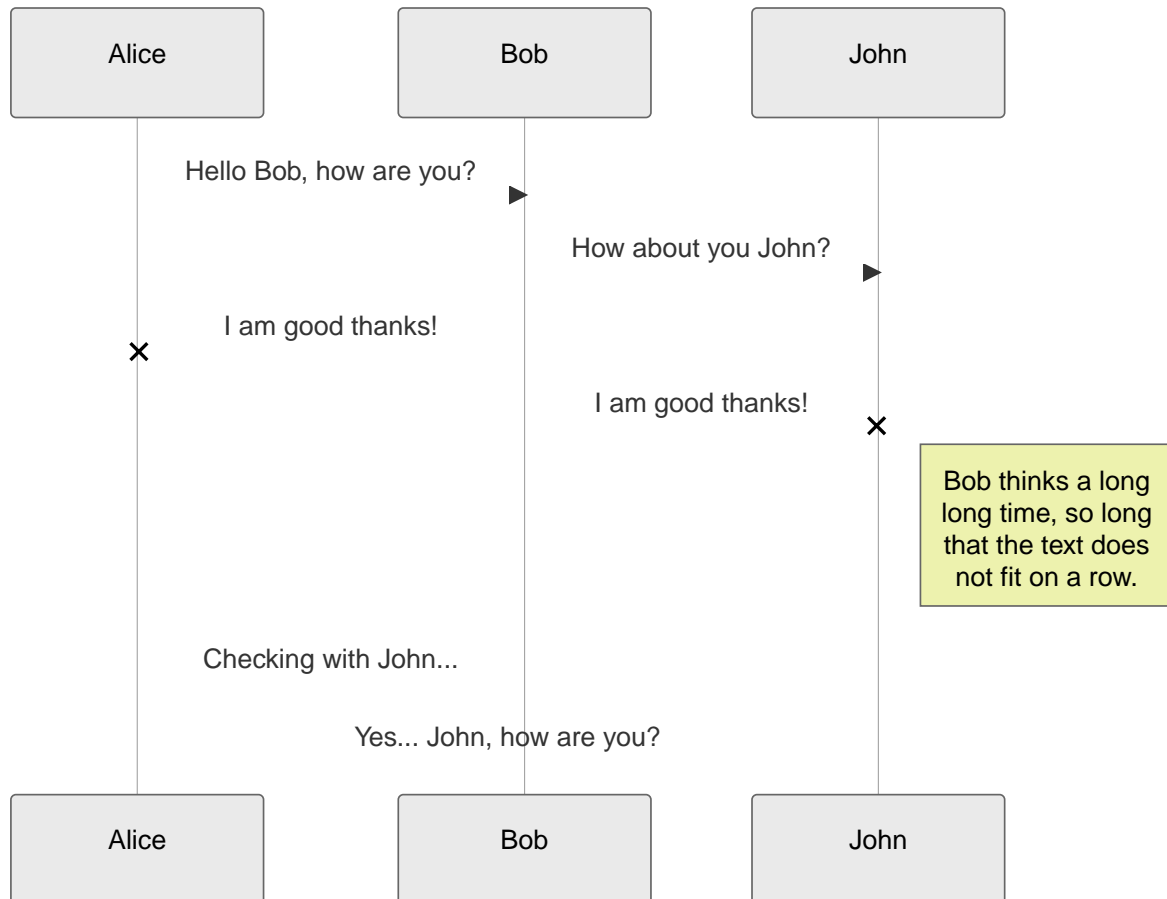


Chapter 14. mermaid

14.1. Internal diagram source



14.2. External diagram source file



Chapter 15. mscgen

Mscgen is a small program that parses Message Sequence Chart descriptions and produces PNG, SVG, EPS or server side image maps (ismaps) as the output. Message Sequence Charts (MSCs) are a way of representing entities and interactions over some time period and are often used in combination with SDL.

— mscgen, <http://www.mcternan.me.uk/mscgen/>

15.1. Internal diagram source



The `mscgen` backend is currently not supported.

15.2. External diagram source file



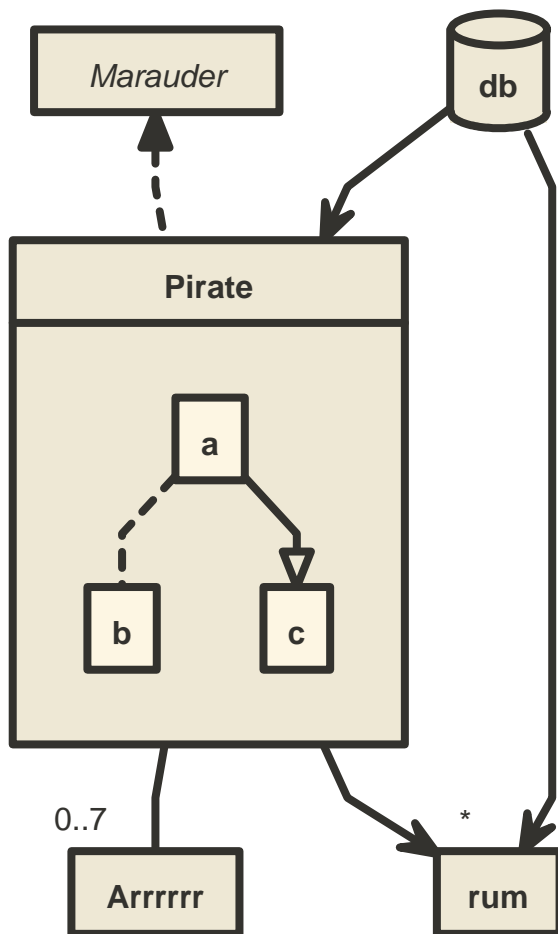
The `mscgen` backend is currently not supported.

Chapter 16. Nomnoml

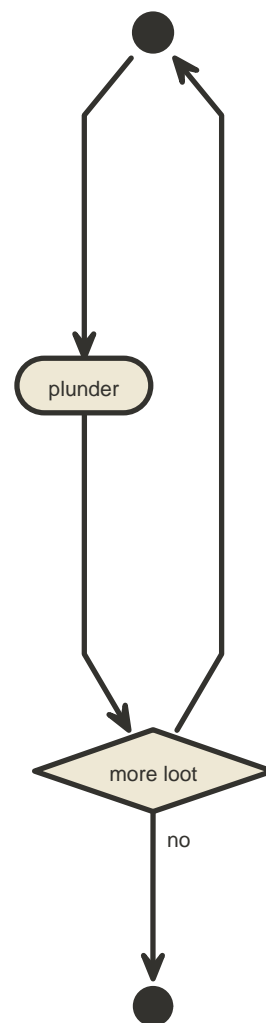
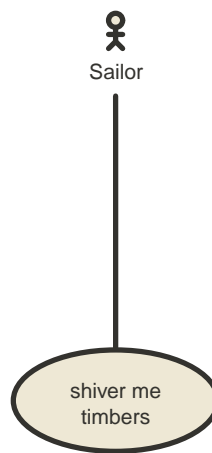
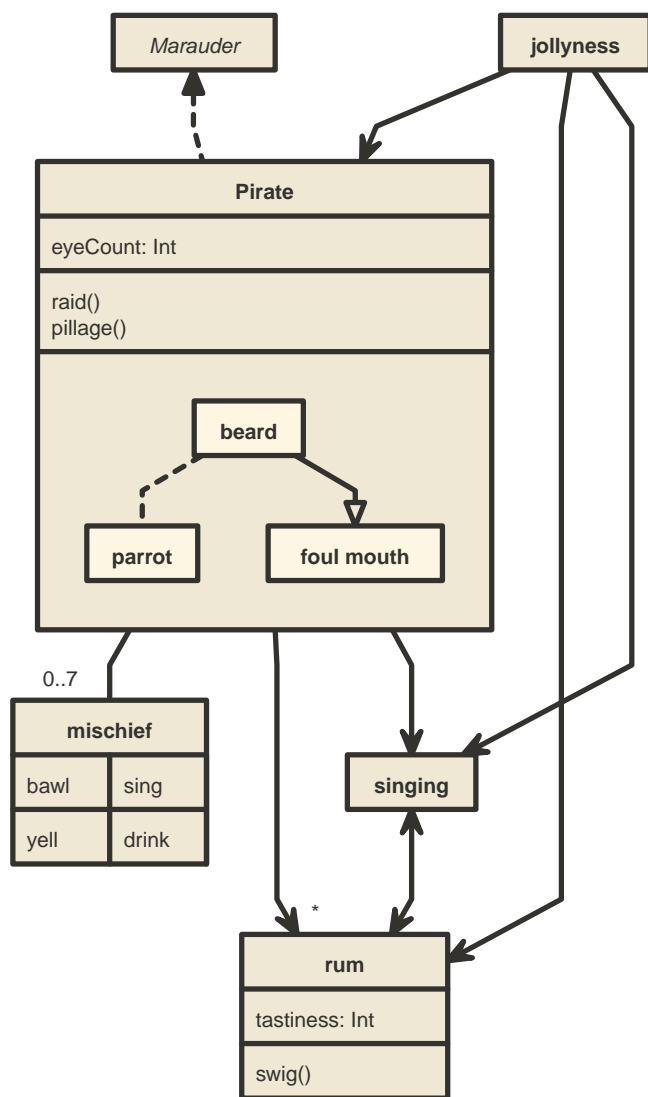
Nomnoml is a tool for drawing UML diagrams based on a simple syntax. It tries to keep its syntax visually as close as possible to the generated UML diagram without resorting to ASCII drawings.

— Daniel Kallin, <https://github.com/skanaar/nomnoml>

16.1. Internal diagram source



16.2. External diagram source file

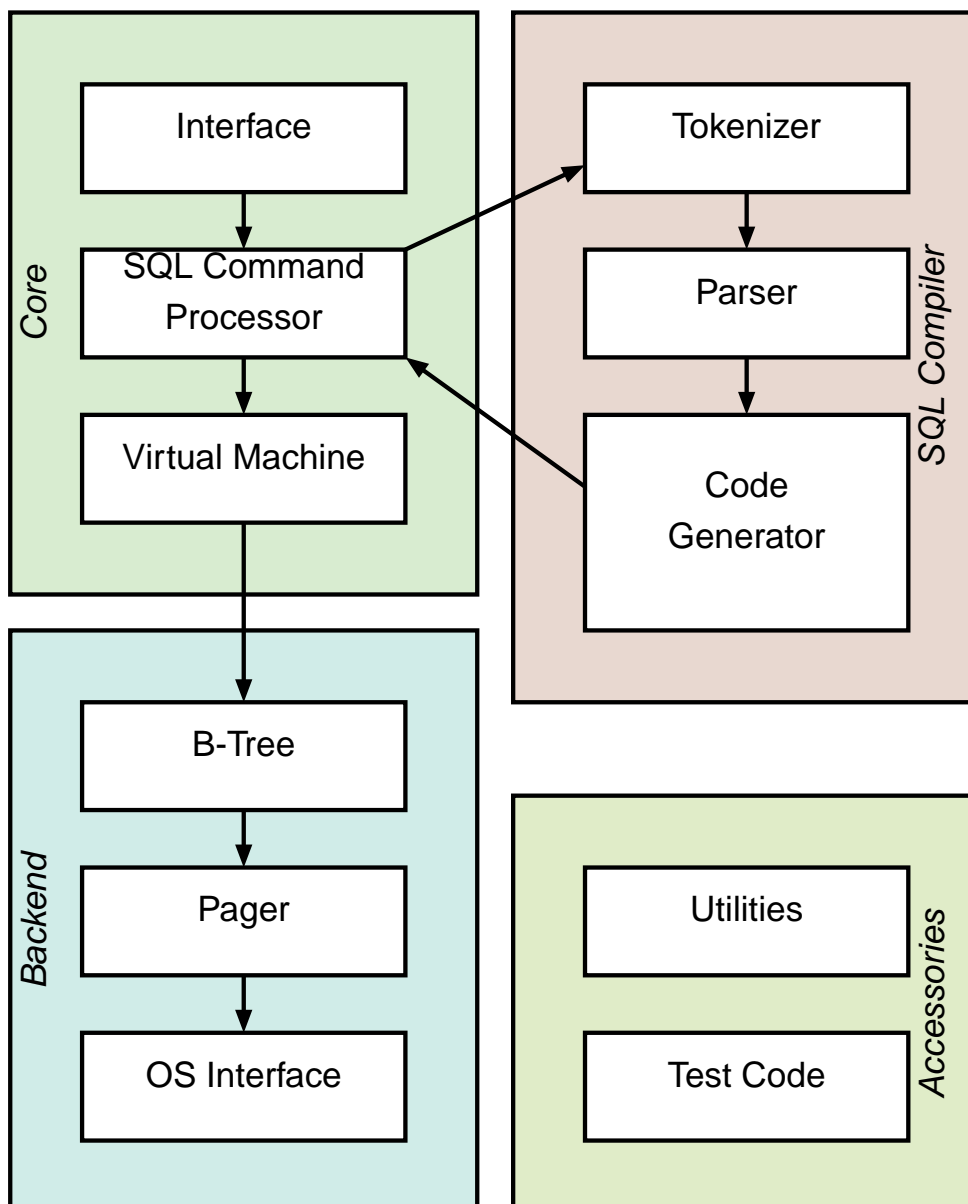


Chapter 17. Pikchr

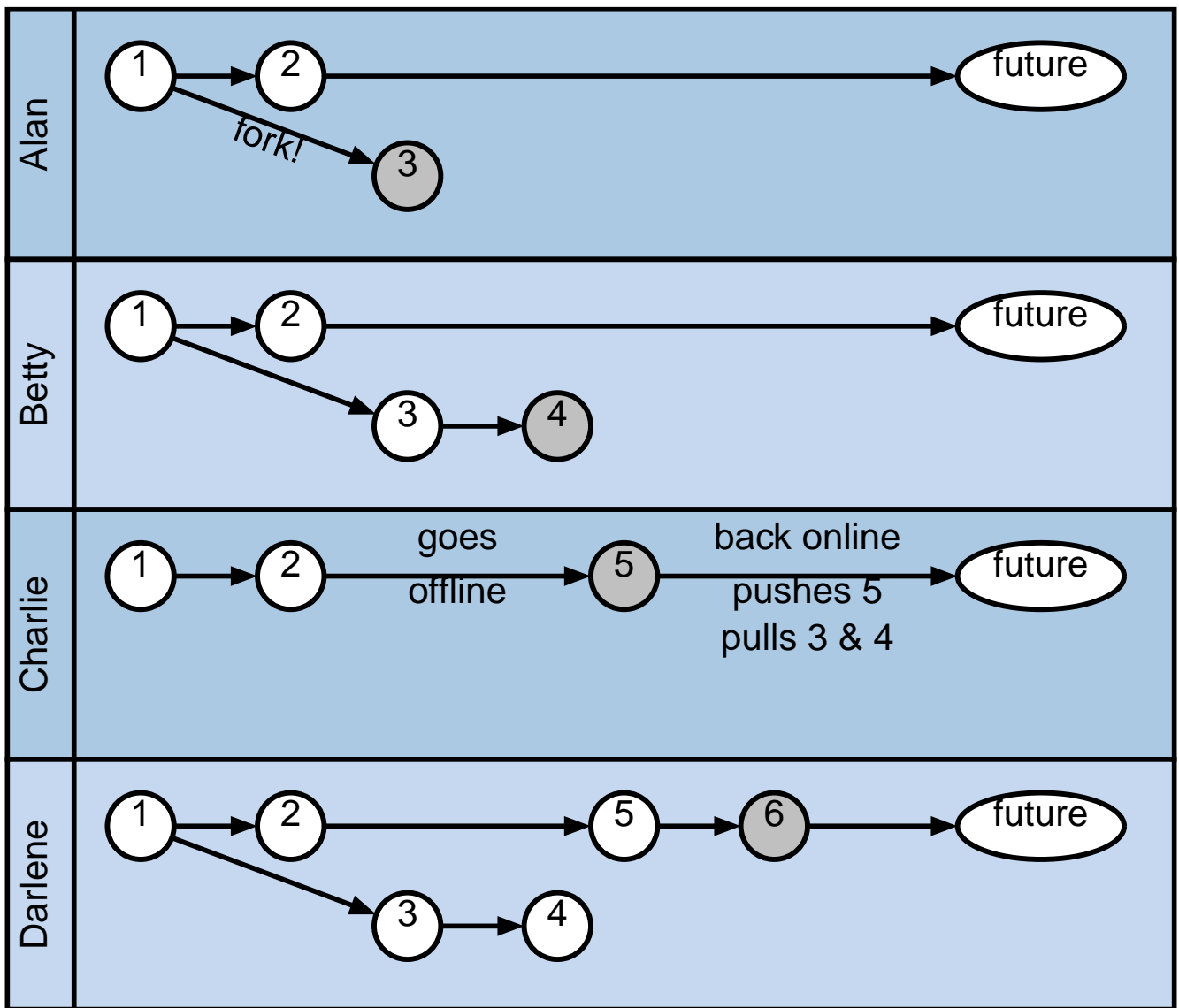
Pikchr (pronounced "picture") is a PIC-like markup language for diagrams in technical documentation. Pikchr is designed to be embedded in fenced code blocks of Markdown or similar mechanisms of other documentation markup languages.

— Pikchr, <https://pikchr.org/home/doc/trunk/homepage.md>

17.1. Internal diagram source



17.2. External diagram source file



Chapter 18. PlantUML

PlantUML is a component that allows to quickly write :

- Sequence diagram
- Usecase diagram
- Class diagram
- Activity diagram (here is the legacy syntax)
- Component diagram
- State diagram
- Object diagram
- Deployment diagram
- Timing diagram

The following non-UML diagrams are also supported:

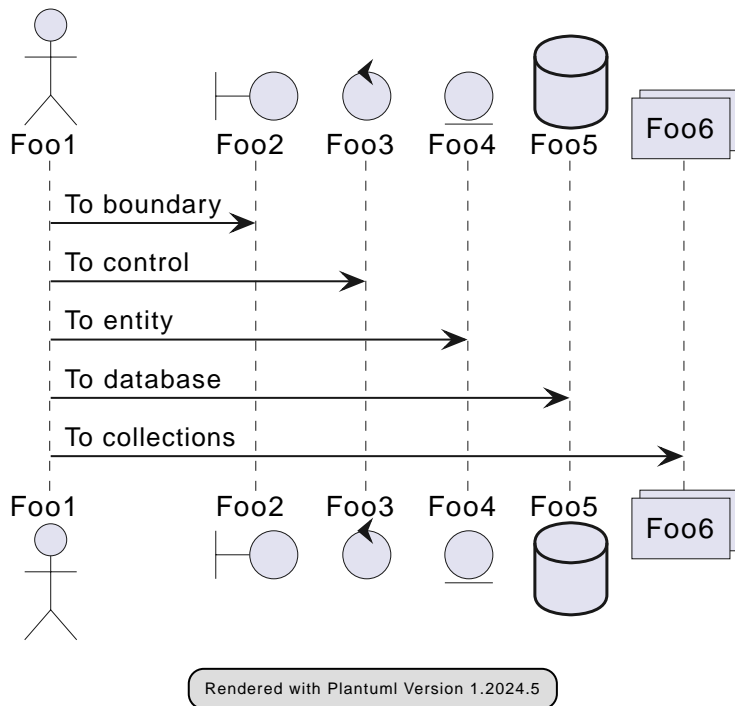
- Network
- Wireframe graphical interface
- Archimate diagram
- Specification and Description Language (SDL)
- Dita diagram
- Gantt diagram
- MindMap diagram
- Work Breakdown Structure diagram
- Mathematic with AsciiMath or JLaTeXMath notation
- Entity Relationship diagram

Diagrams are defined using a simple and intuitive language.

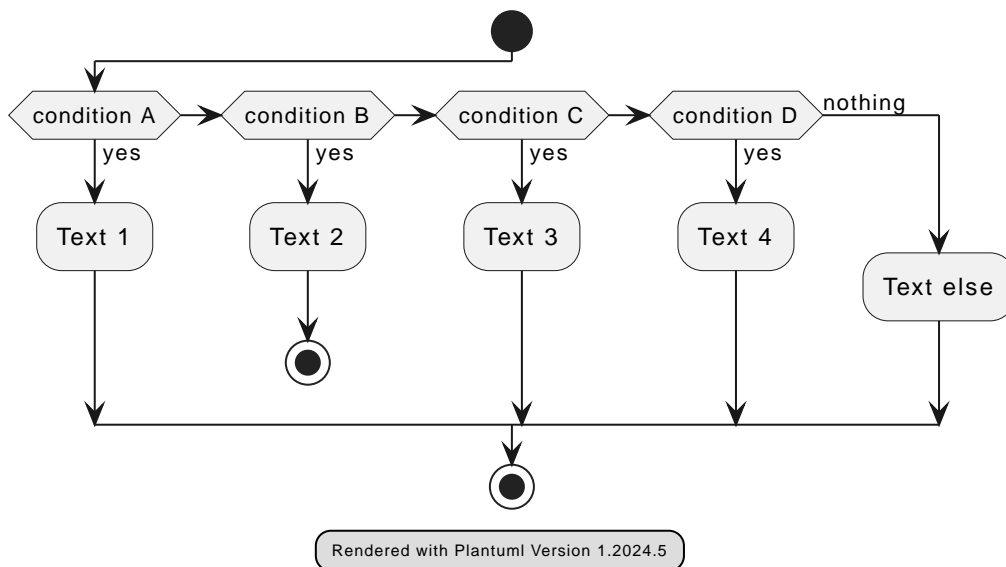
— PlantUML, <https://plantuml.com/>

18.1. PlantUML

18.1.1. Internal diagram source



18.1.2. External diagram source file



18.2. Salt

Salt is a subproject included in PlantUML that may help you to design graphical interface.

— PlantUML, <https://plantuml.com/salt>

18.2.1. Internal diagram source

Just plain text

This is my button

☐ Unchecked radio

☒ Checked radio

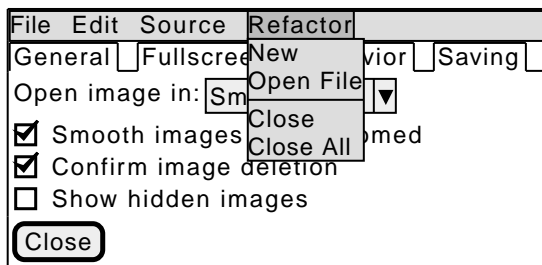
☐ Unchecked box

☒ Checked box

Enter text here

This is a droplist

18.2.2. External diagram source file



Chapter 19. state-machine-cat (smcat)

State Machine cat

write beautiful state charts

— Sander Verweij, <https://github.com/sverweij/state-machine-cat>

19.1. Internal diagram source



19.2. External diagram source file

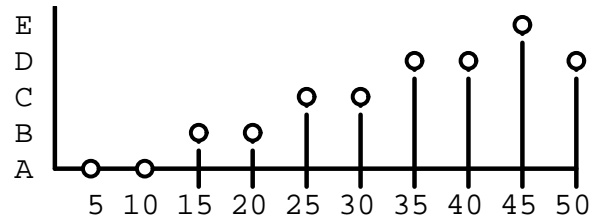
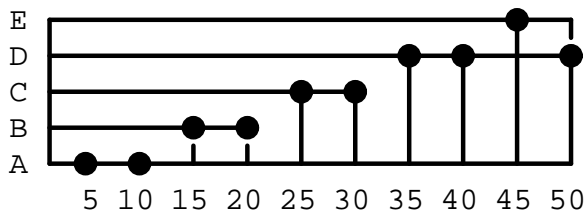


Chapter 20. Svgbob

Svgbob can create a nice graphical representation of your text diagrams.

— Jovansonlee Cesar, <https://github.com/ivanceras/svgbob/>

20.1. Internal diagram source



20.2. External diagram source file

Chapter 21. Symbolator

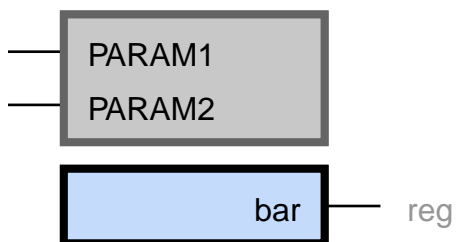
Symbolator is a component diagramming tool for VHDL and Verilog. It will parse HDL source files, extract components or modules and render them as an image.

— Kevin Thibedeau, <https://kevinpt.github.io/symbolator>

21.1. Internal diagram source



21.2. External diagram source file



Chapter 22. Syntrax

Syntrax is a railroad diagram generator. It creates a visual illustration of the grammar used for programming languages.

— Kevin Thibedeau, <https://kevinpt.github.io/syntrax>

22.1. Internal diagram source



22.2. External diagram source file



Chapter 23. Tikz

“What is TikZ?”

Basically, it just defines a number of TEX commands that draw graphics.

— Till Tantau, <https://pgf-tikz.github.io/pgf/pgfmanual.pdf>

23.1. Internal diagram source

```
Failed to generate image: pdflatex failed: This is pdfTeX, Version 3.141592653-2.6-1.40.26 (TeX Live 2024/Alpine Linux) (preloaded format=pdflatex)
\write18 enabled.
I can't find the format file 'pdflatex.fmt'!
```

```
\usetikzlibrary {3d}
\begin{tikzpicture}[z={({10:10mm}),x={{(-45:5mm)}}]
  \def\wave{
    \draw[fill,thick,fill opacity=.2]
      (0,0) sin (1,1) cos (2,0) sin (3,-1) cos (4,0)
      sin (5,1) cos (6,0) sin (7,-1) cos (8,0)
      sin (9,1) cos (10,0) sin (11,-1) cos (12,0);
    \foreach \shift in {0,4,8}
    {
      \begin{scope}[xshift=\shift cm,thin]
        \draw (.5,0) -- (0.5,0 |- 45:1cm);
        \draw (1,0) -- (1,1);
        \draw (1.5,0) -- (1.5,0 |- 45:1cm);
        \draw (2.5,0) -- (2.5,0 |- -45:1cm);
        \draw (3,0) -- (3,-1);
        \draw (3.5,0) -- (3.5,0 |- -45:1cm);
      \end{scope}
    }
  }
  \begin{scope}[canvas is zy plane at x=0,fill=blue]
    \wave
    \node at (6,-1.5) [transform shape] {magnetic field};
  \end{scope}
  \begin{scope}[canvas is zx plane at y=0,fill=red]
    \draw[help lines] (0,-2) grid (12,2);
    \wave
    \node at (6,1.5) [rotate=180,xscale=-1,transform shape] {electric field};
  \end{scope}
\end{tikzpicture}
```

23.2. External diagram source file

Failed to generate image: pdflatex failed: This is pdfTeX, Version 3.141592653-2.6-1.40.26 (TeX Live 2024/Alpine Linux) (preloaded format=pdflatex)

\write18 enabled.

I can't find the format file 'pdflatex.fmt'!

```
% 3D Cone
% Author: Gene Ressler. Adapted to TikZ by Kjell Magne Fauske.
% See http://www.frontiernet.net/~eugene.ressler/ for more details.
% The following code is generated by Sketch. I have edited it a bit
% to make it easier to read.
\begin{tikzpicture}[join=round]
  \tikzstyle{conefill} = [fill=blue!20,fill opacity=0.8]
  \tikzstyle{ann} = [fill=white,font=\footnotesize,inner sep=1pt]
  \tikzstyle{ghostfill} = [fill=white]
  \tikzstyle{ghostdraw} = [draw=black!50]
  \filldraw[conefill](-.775,1.922)--(-1.162,.283)--(-.274,.5)
    --(-.183,2.067)--cycle;
  \filldraw[conefill](-.183,2.067)--(-.274,.5)--(.775,.424)
    --(.516,2.016)--cycle;
  \filldraw[conefill](.516,2.016)--(.775,.424)--(1.369,.1)
    --(.913,1.8)--cycle;
  \filldraw[conefill](-.913,1.667)--(-1.369,-.1)--(-1.162,.283)
    --(-.775,1.922)--cycle;
  \draw(1.461,.107)--(1.734,.127);
  \draw[arrows=<->](1.643,1.853)--(1.643,.12);
  \filldraw[conefill](.913,1.8)--(1.369,.1)--(1.162,-.283)
    --(.775,1.545)--cycle;
  \draw[arrows=>,line width=.4pt](.274,-.5)--(0,0)--(0,2.86);
  \draw[arrows=-,line width=.4pt](0,0)--(-1.369,-.1);
  \draw[arrows=>,line width=.4pt](-1.369,-.1)--(-2.1,-.153);
  \filldraw[conefill](-.516,1.45)--(-.775,-.424)--(-1.369,-.1)
    --(-.913,1.667)--cycle;
  \draw(-1.369,.073)--(-1.369,2.76);
  \draw(1.004,1.807)--(1.734,1.86);
  \filldraw[conefill](.775,1.545)--(1.162,-.283)--(.274,-.5)
    --(.183,1.4)--cycle;
  \draw[arrows=<->](0,2.34)--(-.913,2.273);
  \draw(-.913,1.84)--(-.913,2.447);
  \draw[arrows=<->](0,2.687)--(-1.369,2.587);
  \filldraw[conefill](.183,1.4)--(.274,-.5)--(-.775,-.424)
    --(-.516,1.45)--cycle;
  \draw[arrows=<-,line width=.4pt](.42,-.767)--(.274,-.5);
  \node[ann] at (-.456,2.307) {$r_0$};
  \node[ann] at (-.685,2.637) {$r_1$};
  \node[ann] at (1.643,.987) {$h$};
  \path (.42,-.767) node[below] {$x$}
    (0,2.86) node[above] {$y$}
    (-2.1,-.153) node[left] {$z$};
```

```

% Second version of the cone
\begin{scope}[xshift=3.5cm]
\filldraw[ghostdraw,ghostfill](-.775,1.922)--(-1.162,.283)--(-.274,.5)
--(-.183,2.067)--cycle;
\filldraw[ghostdraw,ghostfill](-.183,2.067)--(-.274,.5)--(.775,.424)
--(.516,2.016)--cycle;
\filldraw[ghostdraw,ghostfill](.516,2.016)--(.775,.424)--(1.369,.1)
--(.913,1.8)--cycle;
\filldraw[ghostdraw,ghostfill](-.913,1.667)--(-1.369,-.1)--(-1.162,.283)
--(-.775,1.922)--cycle;
\filldraw[ghostdraw,ghostfill](.913,1.8)--(1.369,.1)--(1.162,-.283)
--(.775,1.545)--cycle;
\filldraw[ghostdraw,ghostfill](-.516,1.45)--(-.775,-.424)--(-1.369,-.1)
--(-.913,1.667)--cycle;
\filldraw[ghostdraw,ghostfill](.775,1.545)--(1.162,-.283)--(.274,-.5)
--(.183,1.4)--cycle;
\filldraw[fill=red,fill opacity=0.5](-.516,1.45)--(-.775,-.424)--(.274,-.5)
--(.183,1.4)--cycle;

\fill(-.775,-.424) circle (2pt);
\fill(.274,-.5) circle (2pt);
\fill(-.516,1.45) circle (2pt);
\fill(.183,1.4) circle (2pt);
\path[font=\footnotesize]
    (.913,1.8) node[right] {$i\hbox{$=$}$0$}
    (1.369,.1) node[right] {$i\hbox{$=$}$1$};
\path[font=\footnotesize]
    (-.645,.513) node[left] {$j$}
    (.228,.45) node[right] {$j\hbox{$+$}$1$};
\draw (-.209,.482)+(-60:.25) [yscale=1.3,->] arc(-60:240:.25);
\fill[black,font=\footnotesize]
    (-.516,1.45) node [above] {$P_{00}$}
    (-.775,-.424) node [below] {$P_{10}$}
    (.183,1.4) node [above] {$P_{01}$}
    (.274,-.5) node [below] {$P_{11}$};

\end{scope}
\end{tikzpicture}

```

Chapter 24. UMLet

UMLet is a free, open-source UML tool with a simple user interface: draw UML diagrams fast, create sequence and activity diagrams from plain text, share via exports to eps, pdf, jpg, svg, and clipboard, and develop new, custom UML elements.

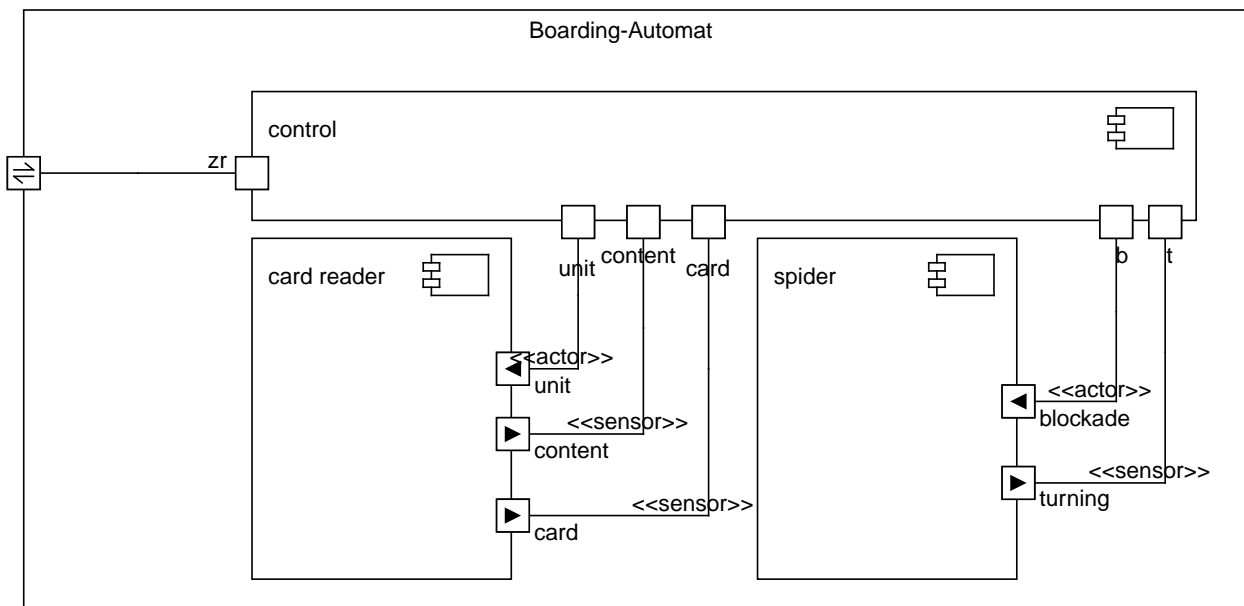
— The UMLet Team, <https://www.umlet.com>

24.1. Internal diagram source



The **umlet** backend currently does not support internal diagram sources.

24.2. External diagram source file

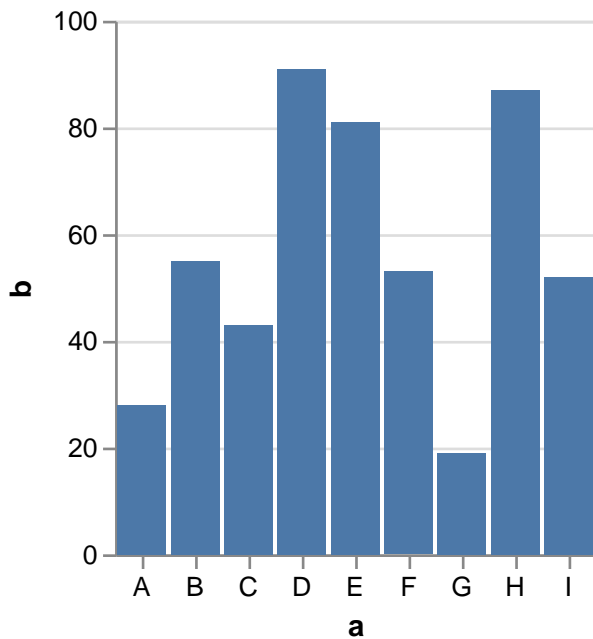


Chapter 25. Vega Lite

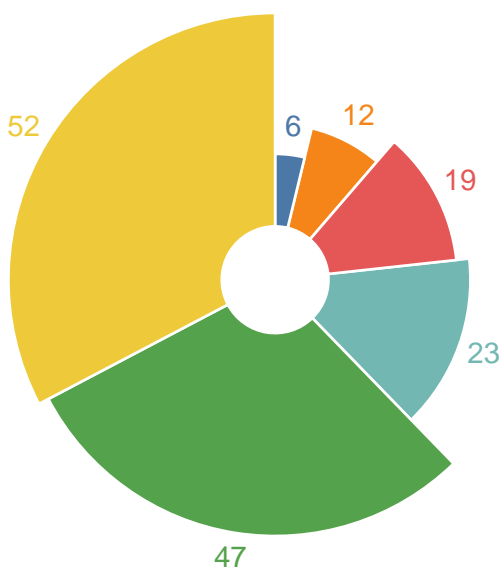
Vega-Lite is a high-level grammar of interactive graphics. It provides a concise, declarative JSON syntax to create an expressive range of visualizations for data analysis and presentation.

— Vega, <https://vega.github.io/vega-lite/>

25.1. Internal diagram source



25.2. External diagram source file

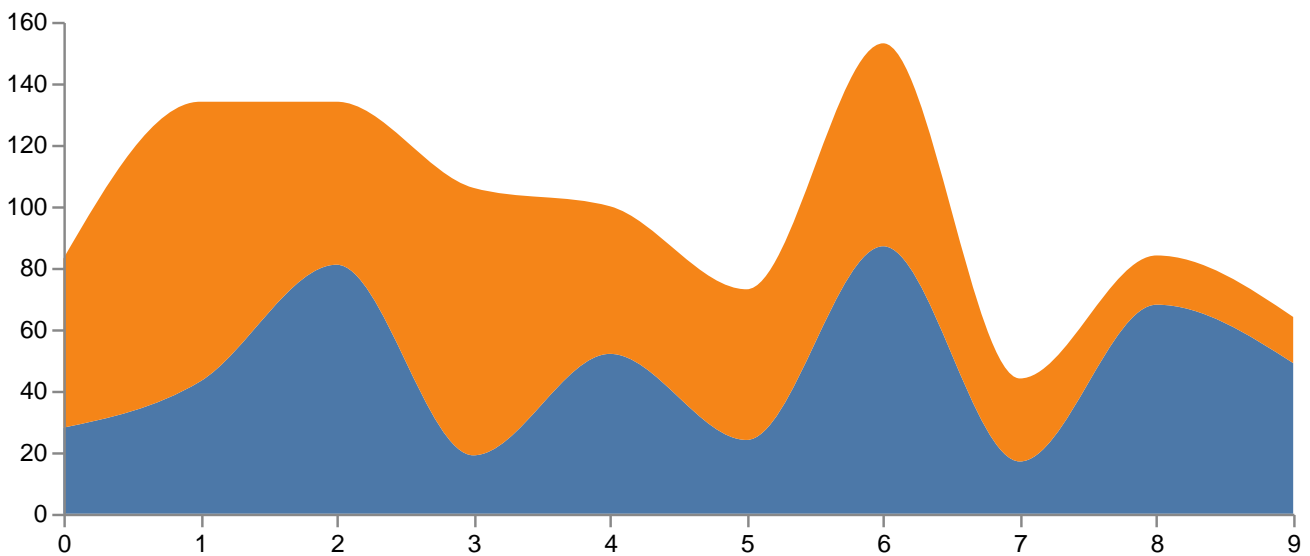


Chapter 26. Vega

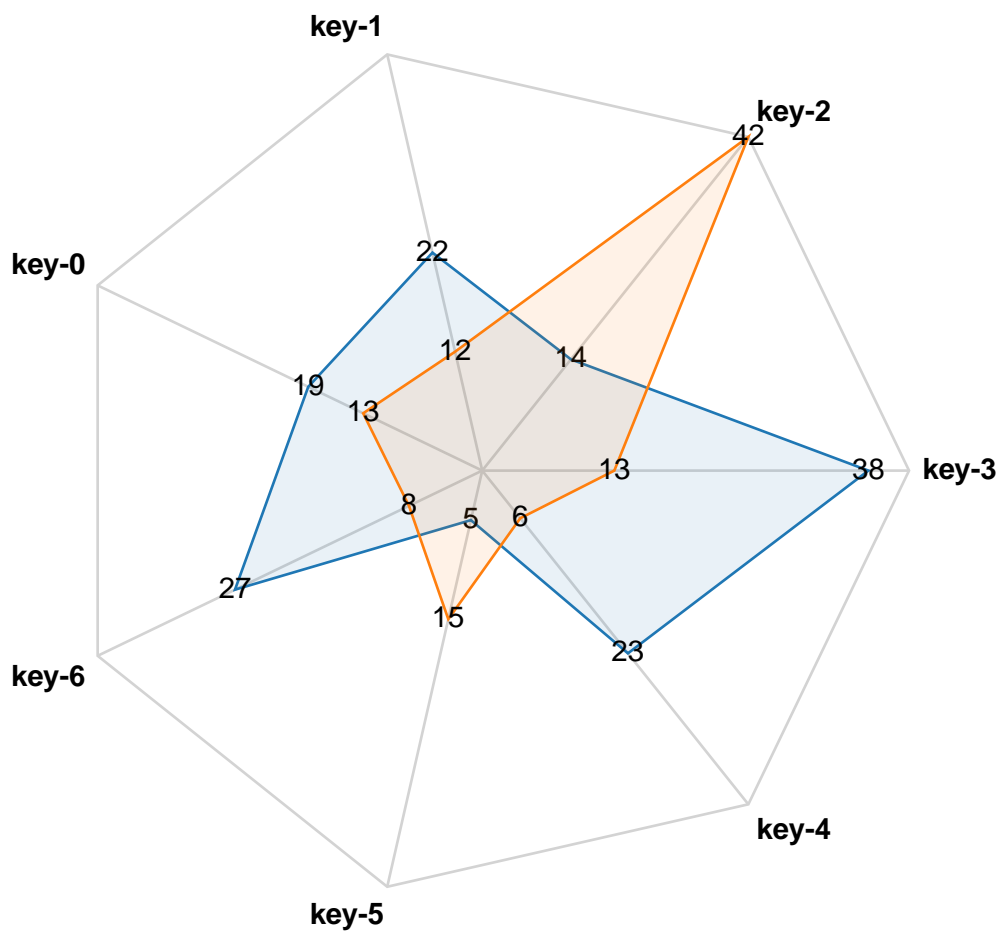
Vega is a visualization grammar, a declarative language for creating, saving, and sharing interactive visualization designs. With Vega, you can describe the visual appearance and interactive behavior of a visualization in a JSON format, and generate web-based views using Canvas or SVG.

— Vega, <https://vega.github.io/vega/>

26.1. Internal diagram source



26.2. External diagram source file

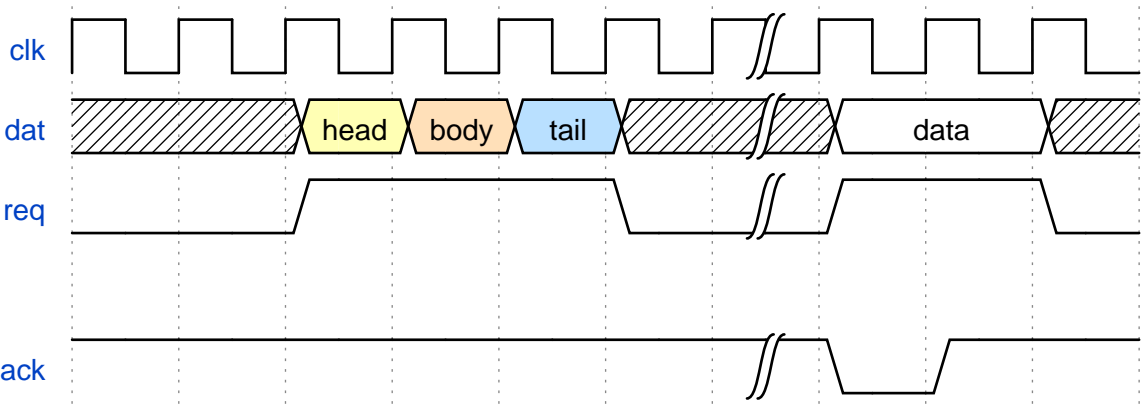


Chapter 27. WaveDrom

WaveDrom draws your Timing Diagram or Waveform from simple textual description.

— WaveDrom, <https://wavedrom.com/>

27.1. Internal diagram source



27.2. External diagram source file

| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|-----|----|----|----|----|----|-------|----|--------------|-----|---|---------------------|-------|--|-------------------------|----|--|-------------------------------------|---|---|--------------------|---|---|---|---|
| 31 | 29 | 28 | 26 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 | | | | | | | | | | | | | |
| nf | | | mop | | | vm | | | lumop | | | rs1 | | | width | | | vd | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | base address | | | destination of load | | | VLxU, VLE zero-extended | | | VLxU, VLE zero-extended, fault-only | | | VLxU sign-extended | | | | |
| | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | |
| | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | |
| | | | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | |

Bibliography

- [DIAG] AsciiDoctor Project (en): *AsciiDoctor Diagram*. <https://asciidoctor.org/docs/asciidoctor-diagram/> (Retrieved March 29, 2020)