Diagram Dojo

Agenda

- Why draw pictures?
- Introduction into DRAKON

Documentation

Long and boring text?

VISMA

If we want to reach a goal, we must understand precisely what we want.
A software project has a chance of success only if the team sees the goal clearly.

Our goal is to make software that fulfills the customer's needs.
We must clearly see what those needs are.
Documentation and specs are unavoidable because of that.

But who likes to read documentation?

Documentation

Long and boring text?

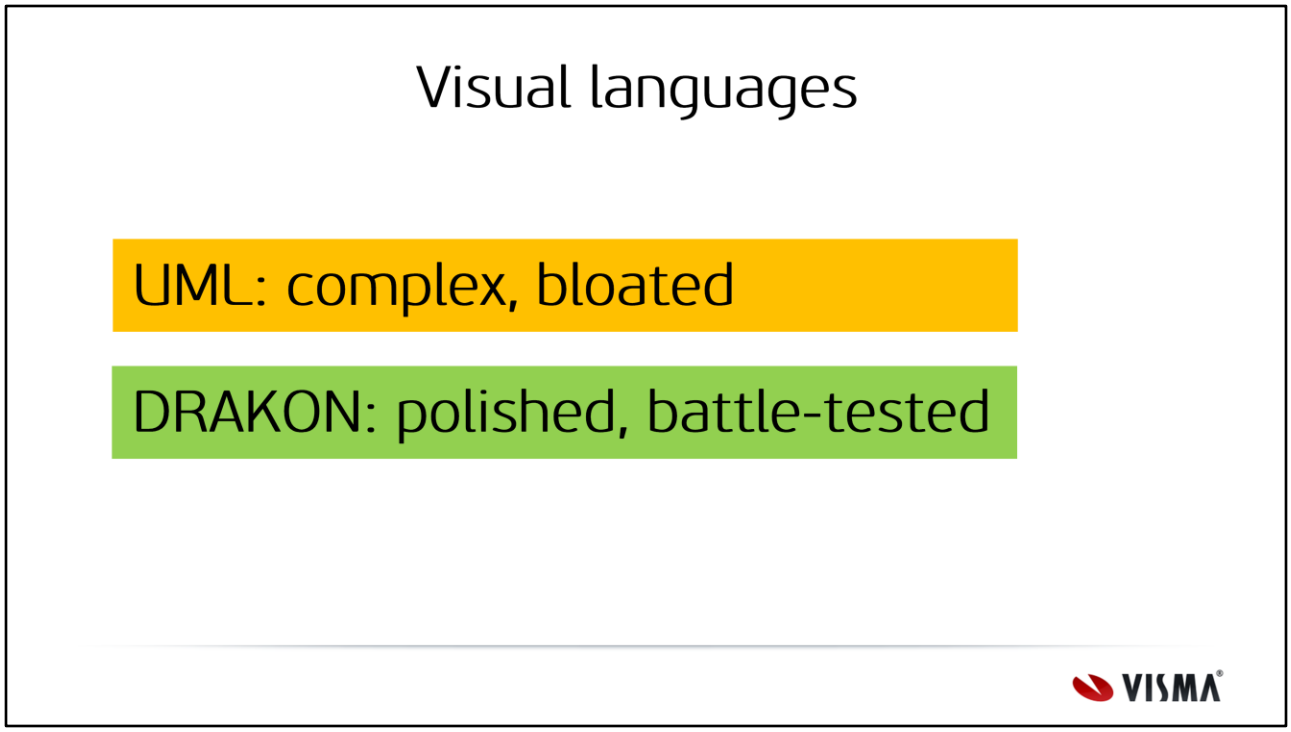
No, thanks

VISMA

Technical text can be hard to read.
This is because our eyes and brain are not optimised for text.
They are optimised for graphics.

Replacing text with graphics is key to leverage our cognitive abilities.
But how to create high-quality graphics quickly in industrial quantities?

Visual languages provide a way to produce graphics in a fast and consistent way.



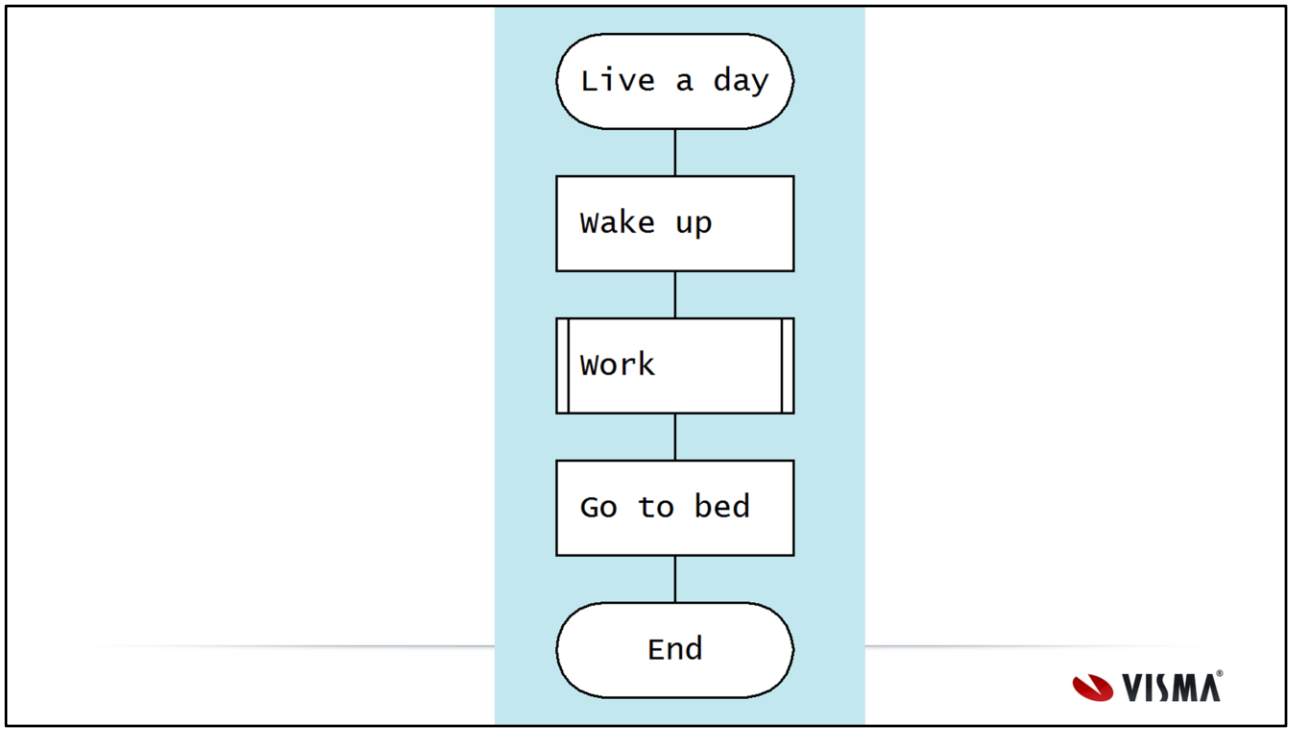
The most famous visual language for software specification is UML.
It is also widely criticised for its complexity and obscurity.

There are others, for example BPMN.

DRAKON is a better alternative to UML.
DRAKON has been carefully designed specifically to be easy to understand.

DRAKON stands behind many success stories in the Russian space program.
The original goal of DRAKON was to explain rocket science to software developers.

But nowadays, DRAKON is used in other areas, including law and health care.

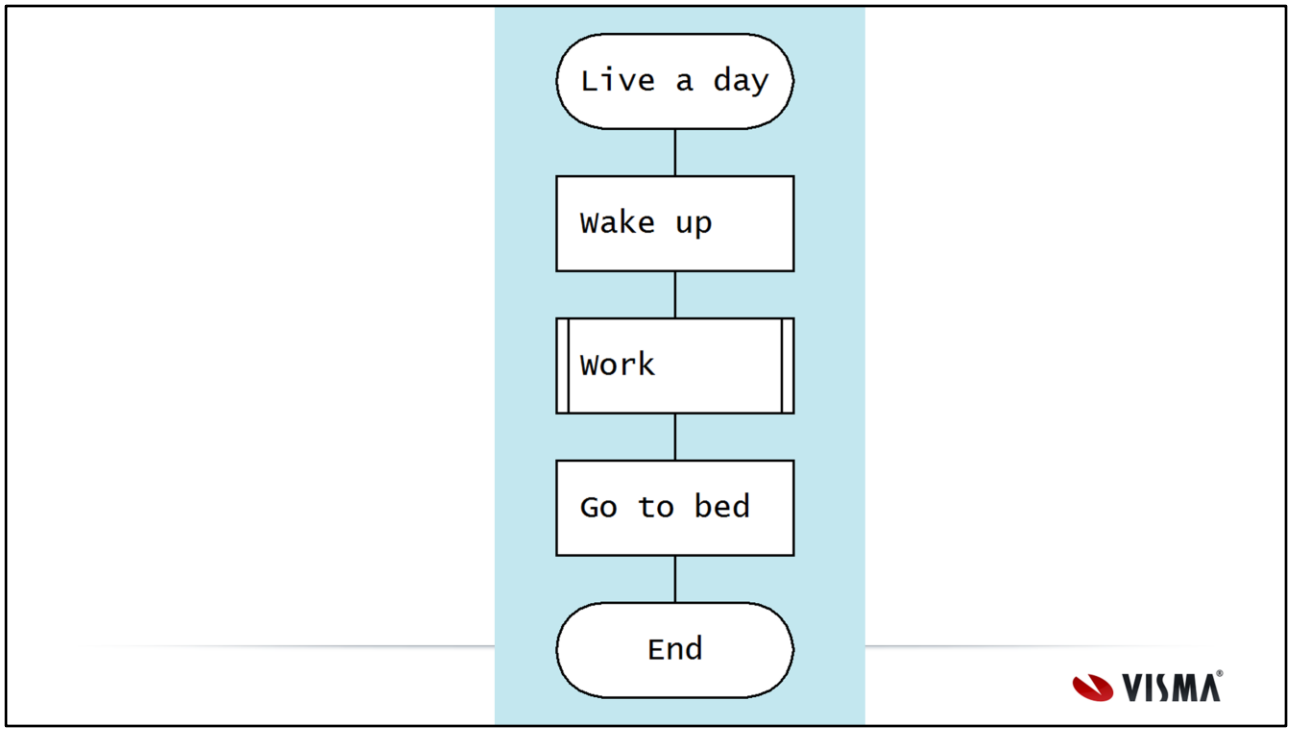


Here is a very simple DRAKON diagram.
It has a header, a body and an end.

With DRAKON, the reader does not need to search for the next step on the diagram.
The next step is always below the current one.

We live in a world with gravity.
Moving down is following gravity. This is how all things tend to move.
So our eyes readily accept that the thing below follows the current one.

In addition, we are trained to read in this direction.



The entry point is located at the top of the diagram.
All DRAKON diagrams have exactly one end. The end is located at the bottom.

Reduce visual noise

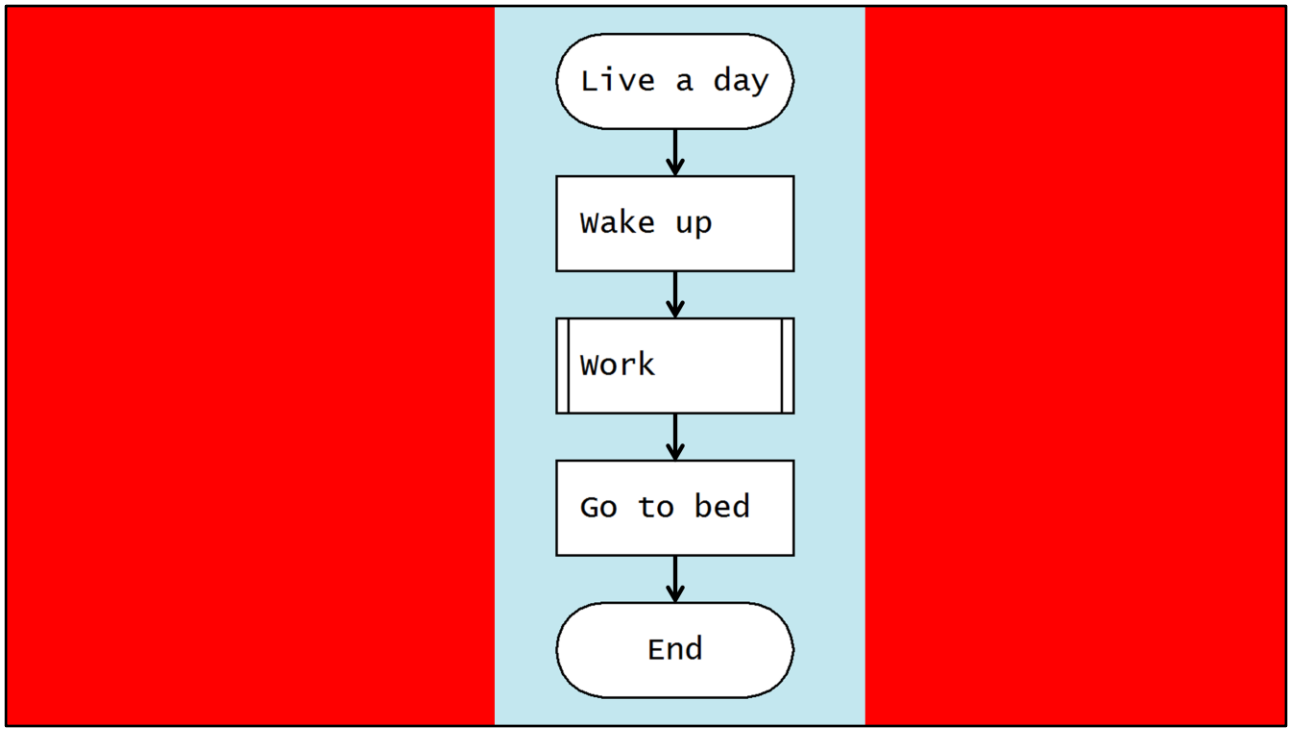
The diagram should include only the necessary information.
All graphical elements that do not carry useful data distract the viewer.

They are often called "chartjunk".

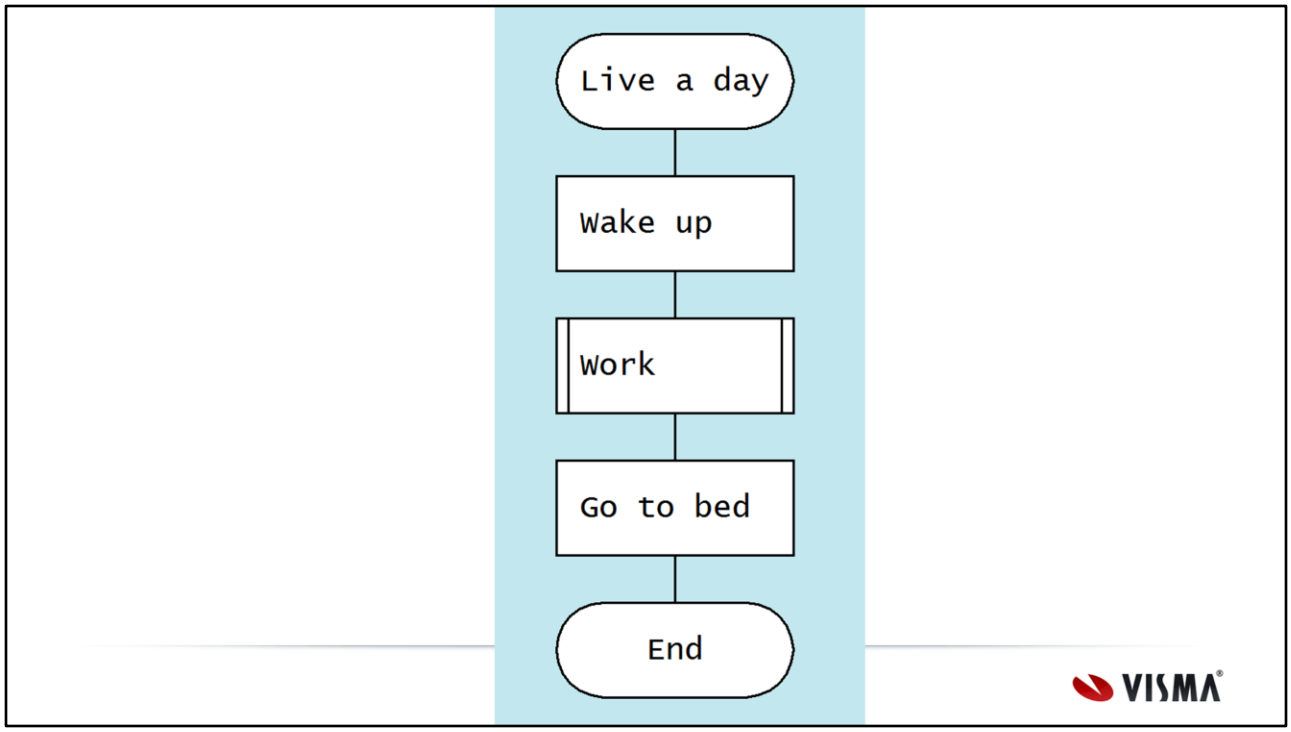
Arrows are a good example.
Do not use arrows to indicate the next step.

The next action is always below.
So arrows are redundant.
They just litter the visual scene.

Actually, DRAKON does use arrows. But they have a special meaning.

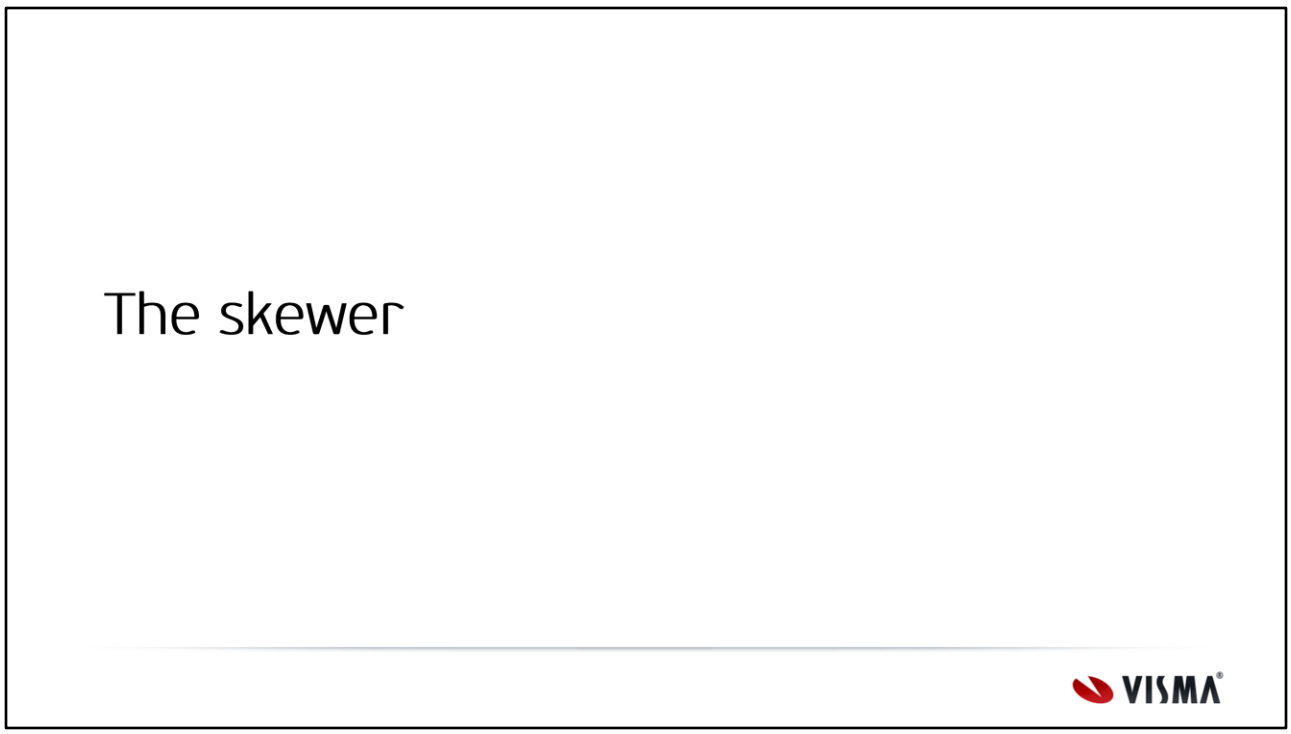


Wrong!
Do not use arrows to point at the next icon.



Right!
Simple, clean lines are enough.

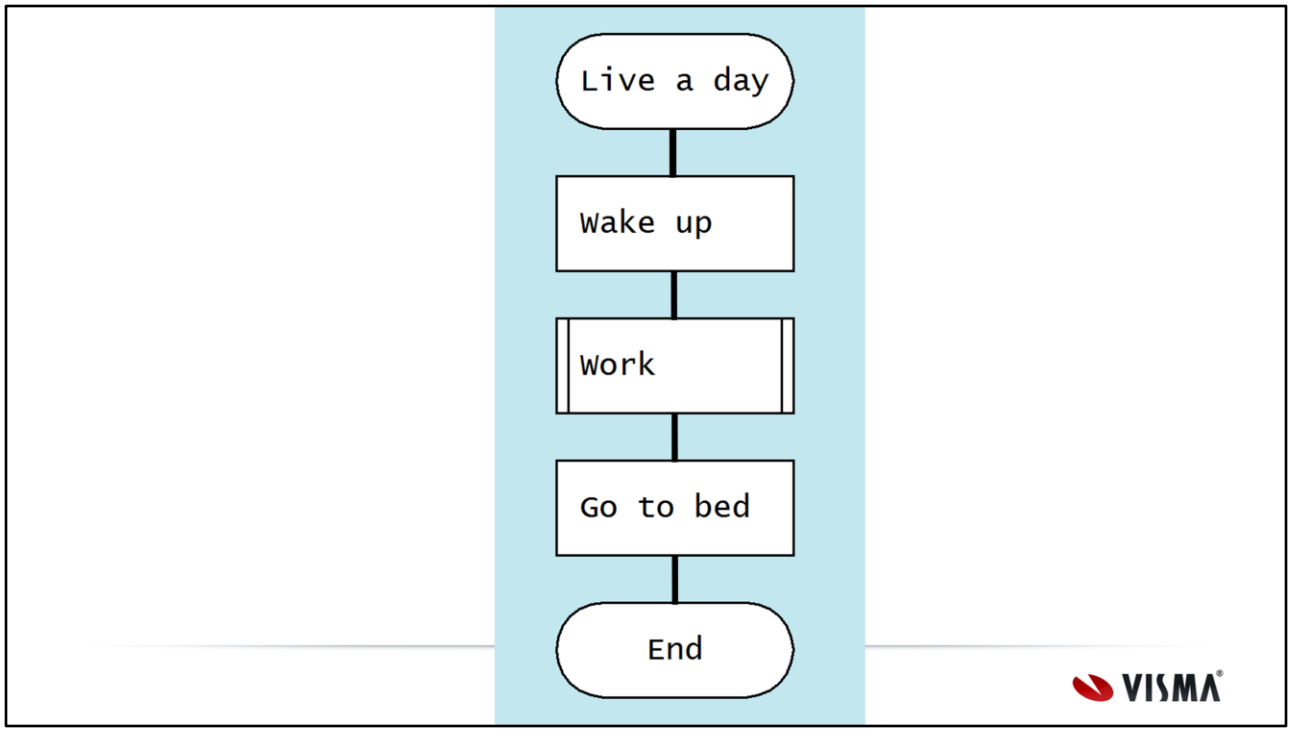
The skewer



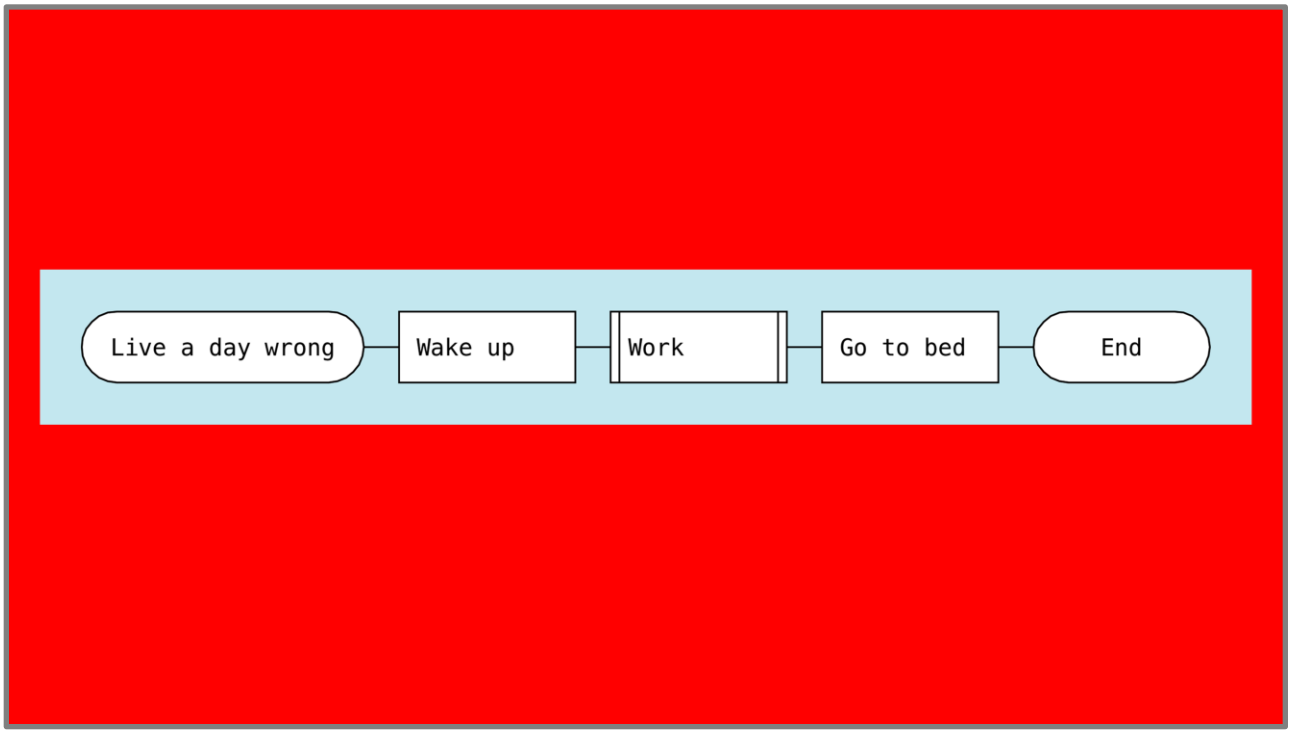
Actions that follow a sequence are connected by a vertical line.
This line is called "the skewer".

The skewer must always be straight and vertical.
It should not go sideways. It should not break or bend.

Turns and unnecessary angles add complexity to the diagram.
We try to avoid all redundant complexity. Our project is already complex enough.

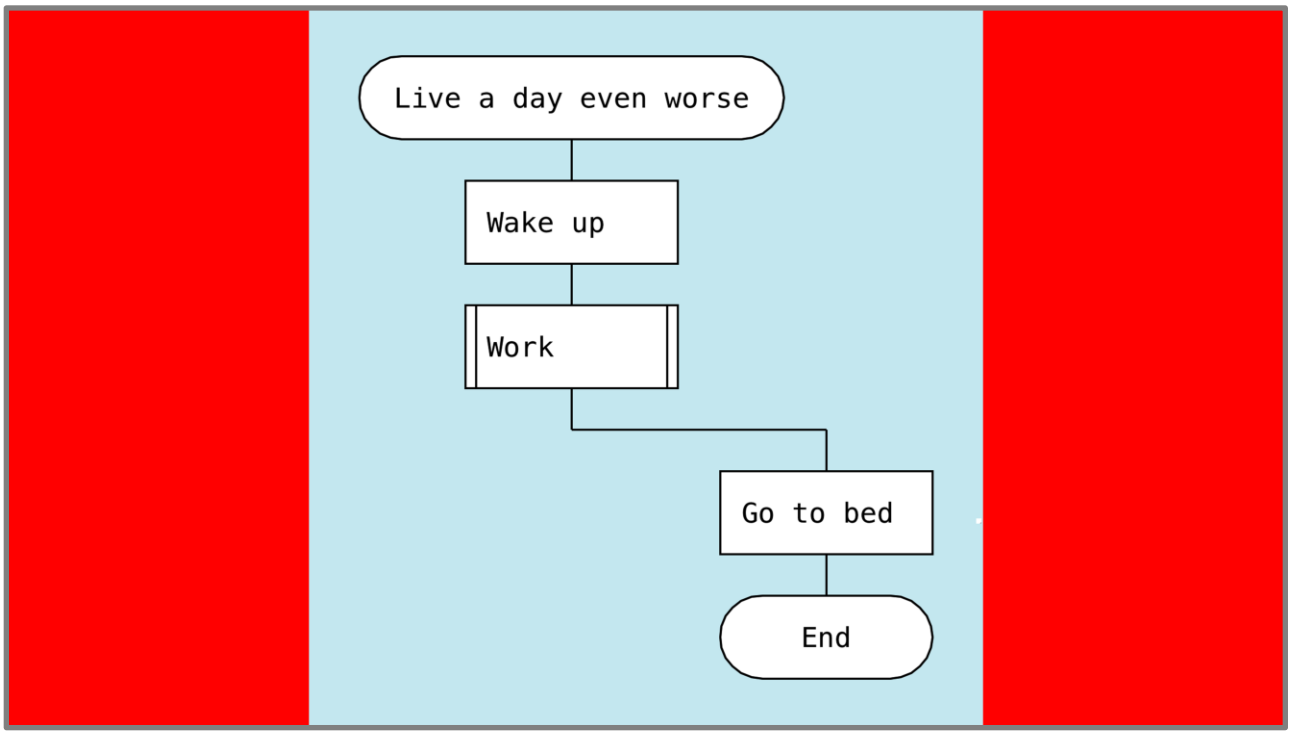


The skewer is an axis that the icons sit on.



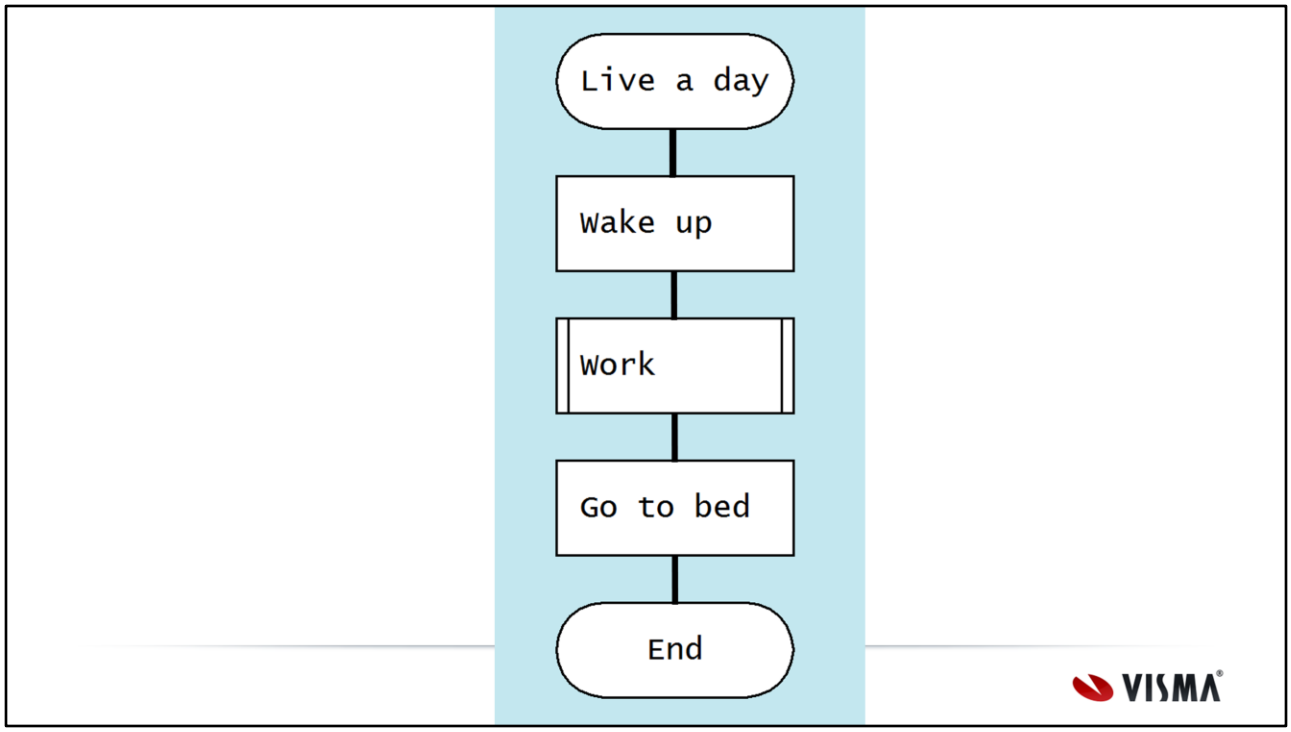
Wrong!

The skewer is not vertical.



Wrong!

The skewer is broken.



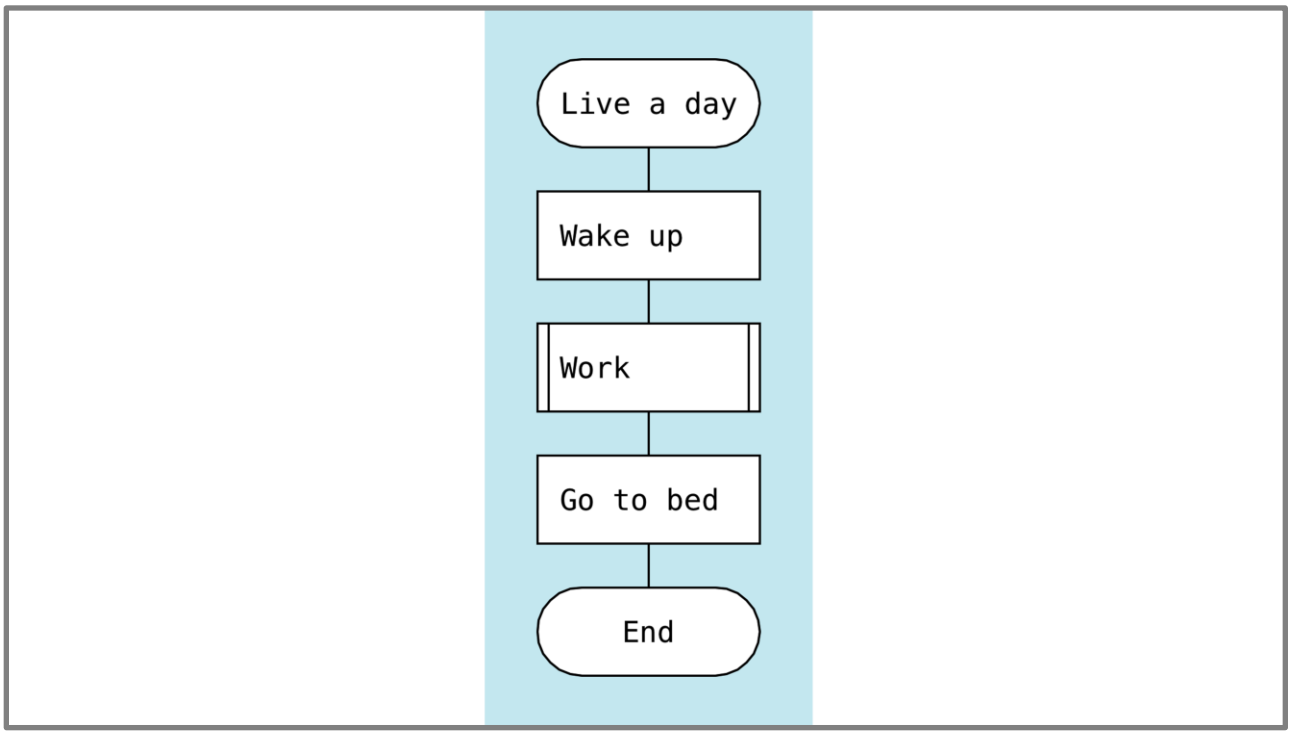
The skewer must be vertical.

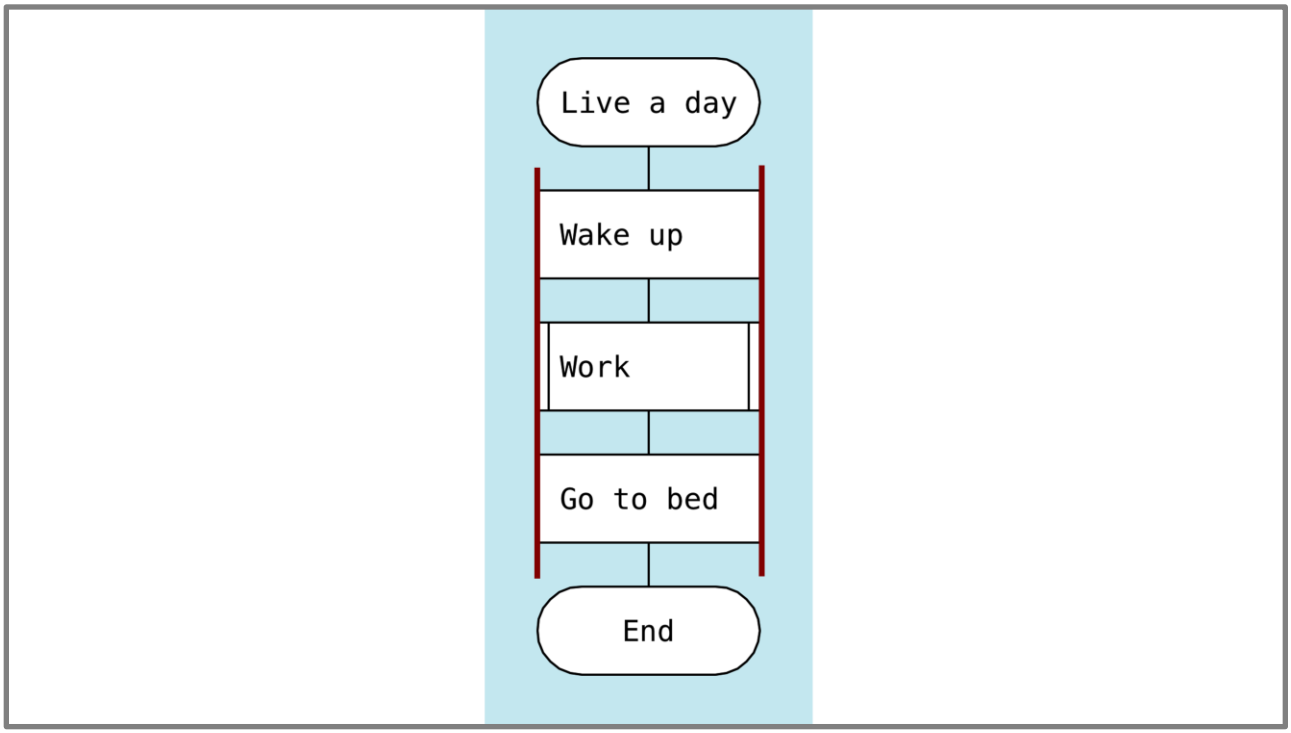
Same width

The icons on the skewer must be centered horizontally and have the same width.
The width requirement does not apply to the header and the end icons.

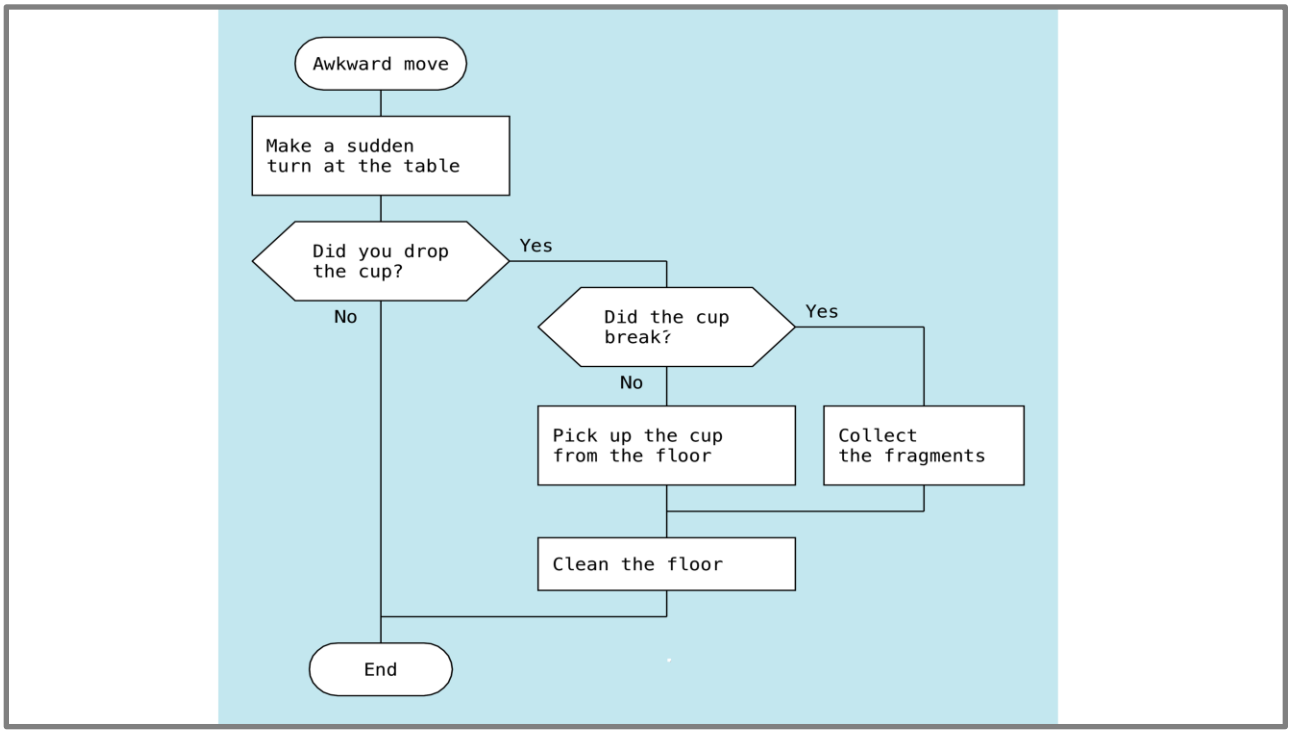
When several elements have aligned boundaries, those elements seem to compose a single entity.
And this is exactly what we want to achieve.

Even a small variation in width immediately attracts attention. Why is that? Why does that icon sticks out?

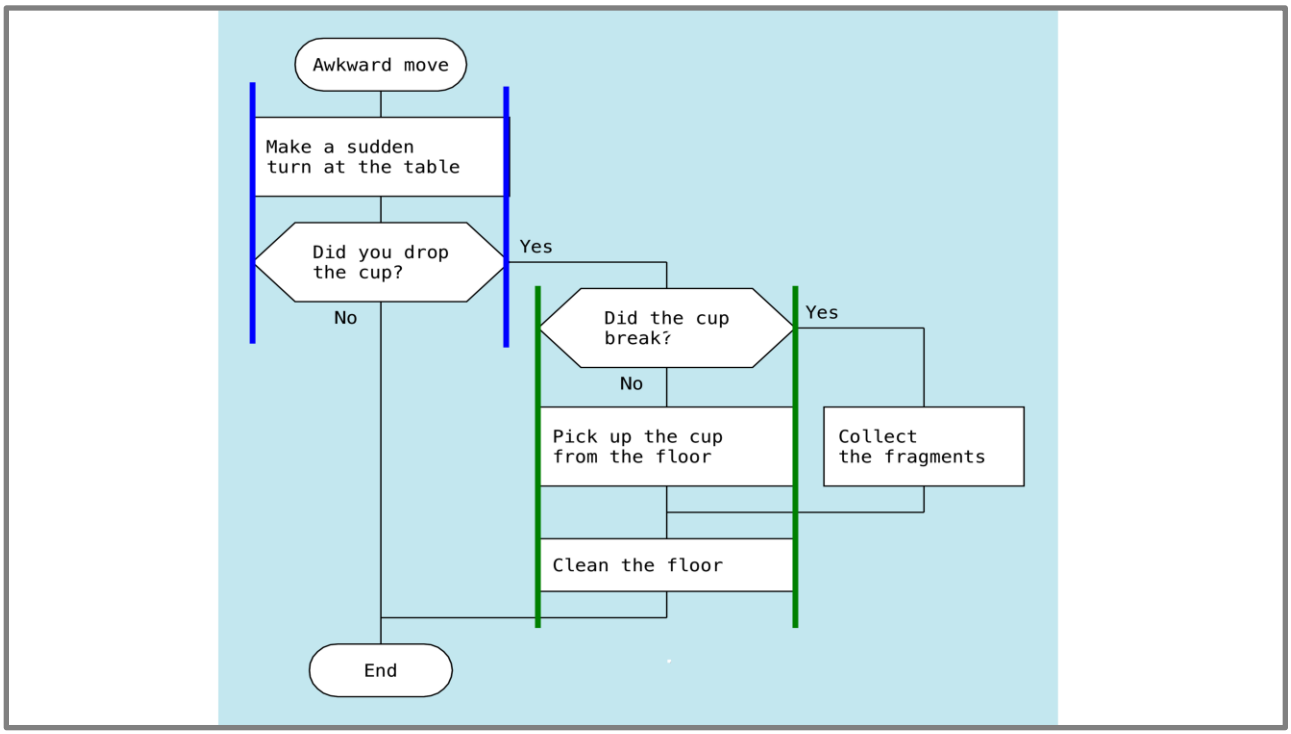

Live a day
Wake up
Work
Go to bed
End



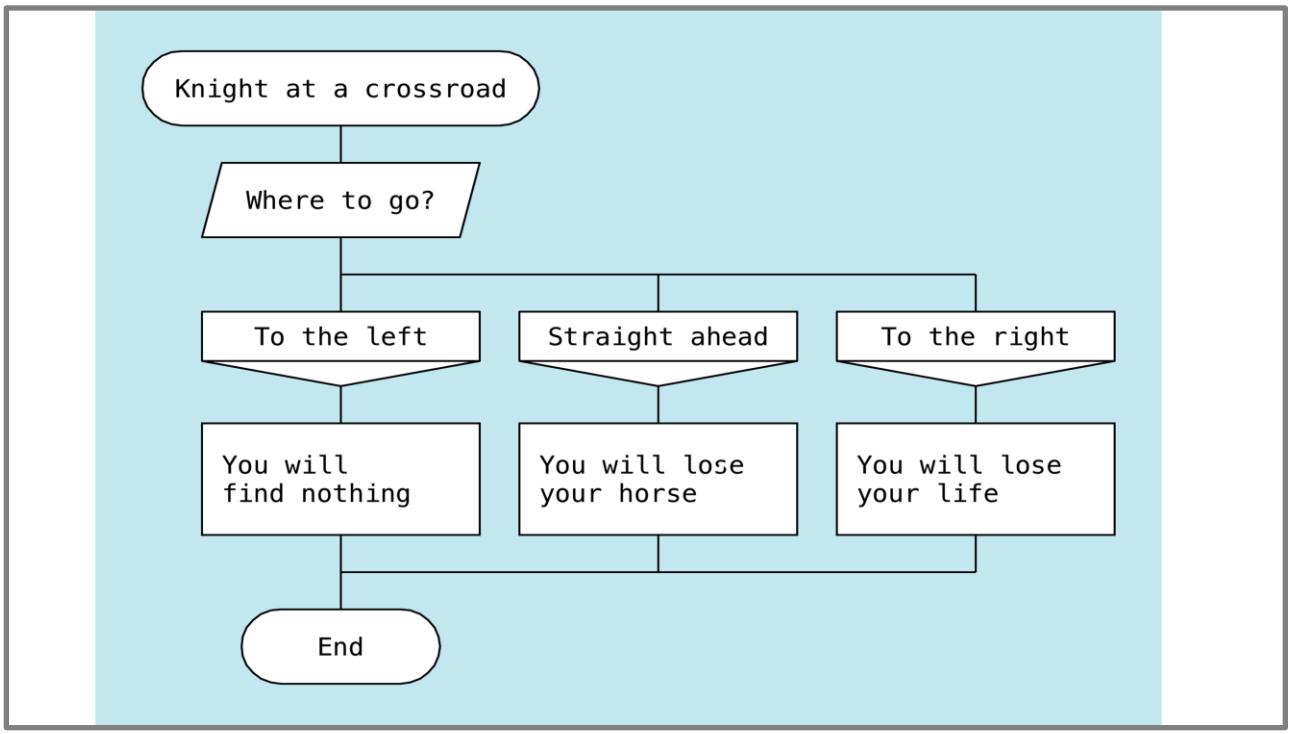
All icons except the header and the end are on the same width.



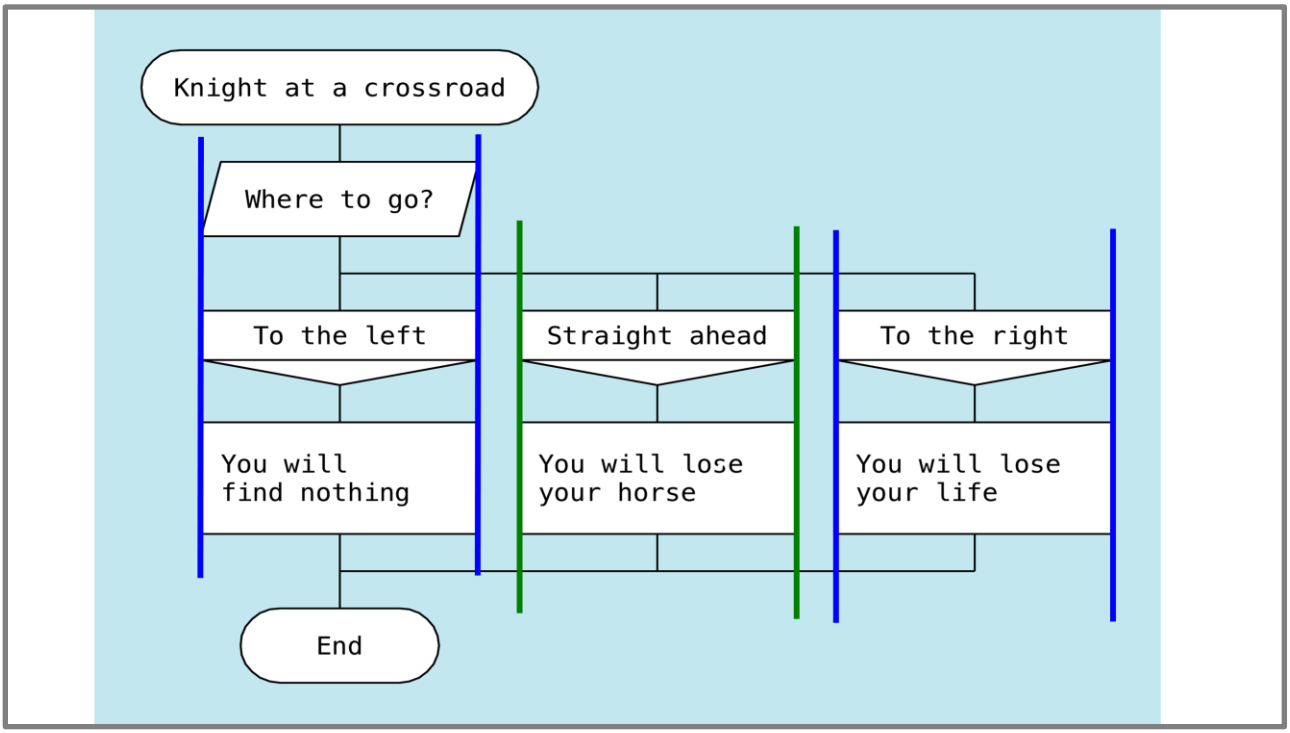
A slightly more complex diagram. It has several verticals.



The rule of the same width still applies.



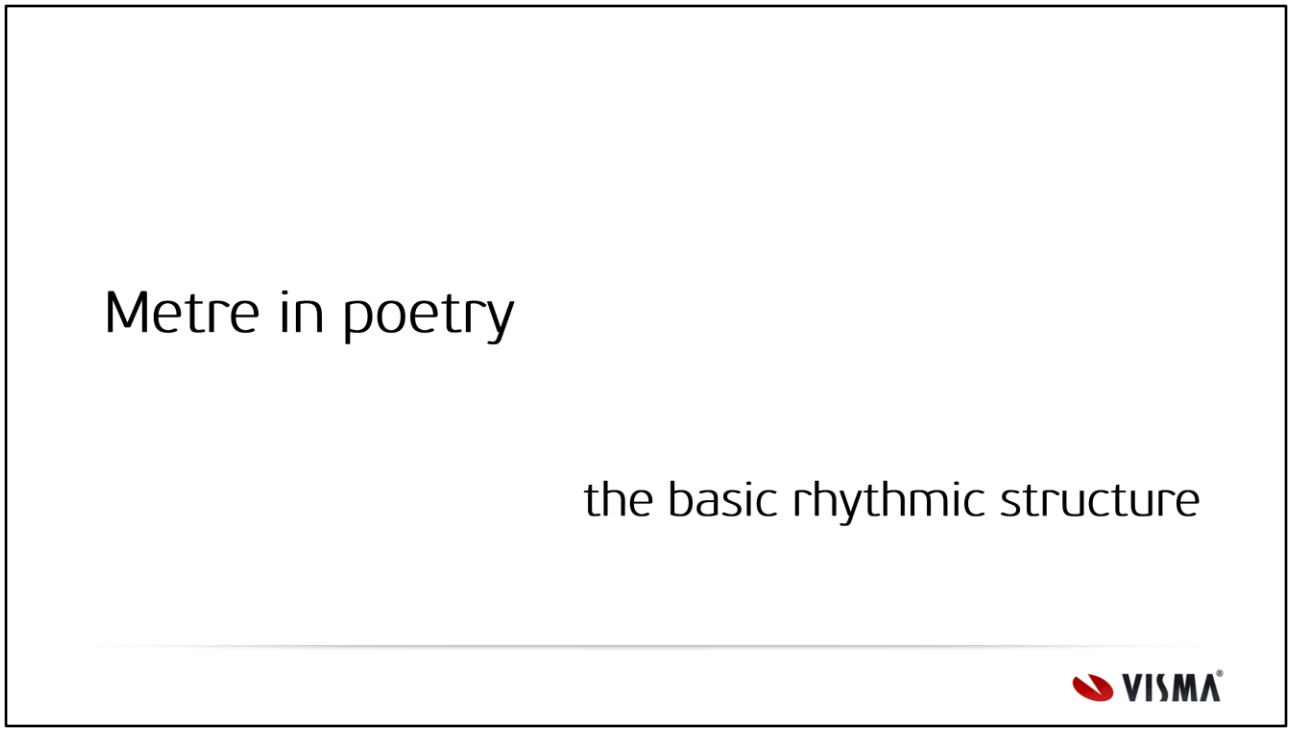
Another example.



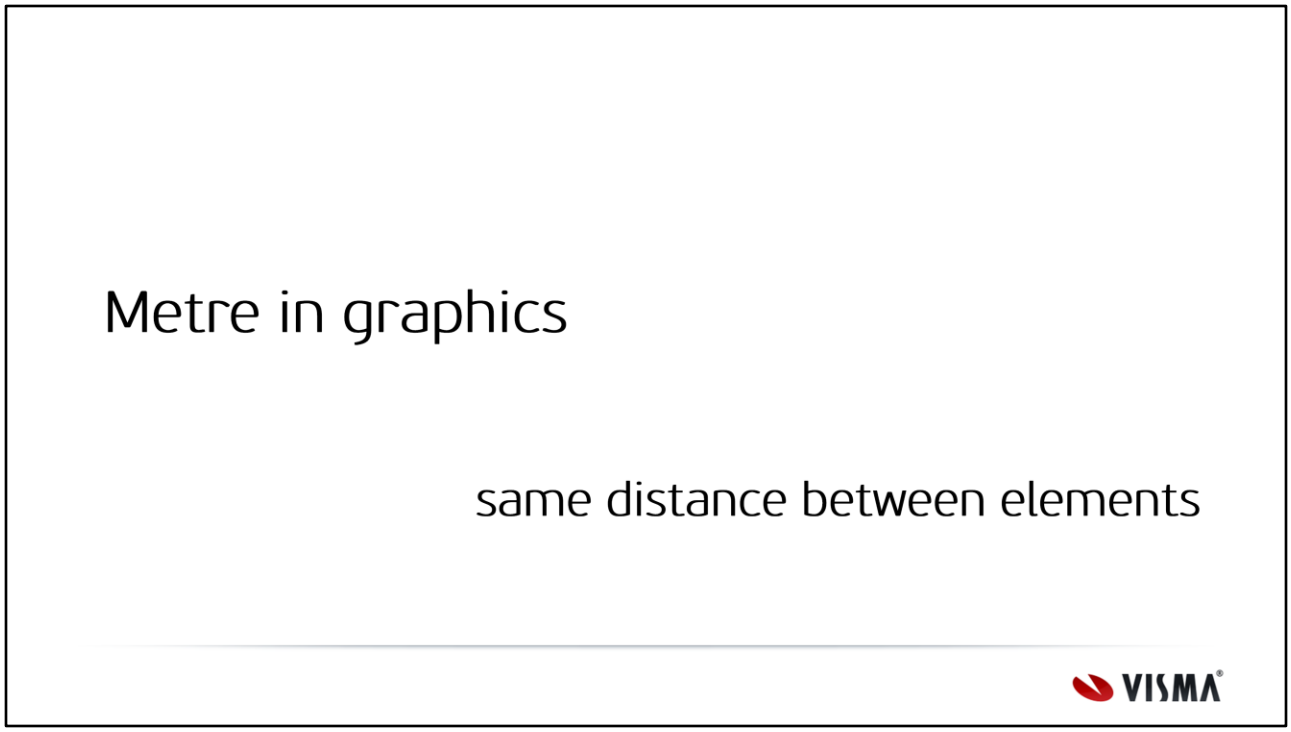
The rule of the same width applies again.

Metre

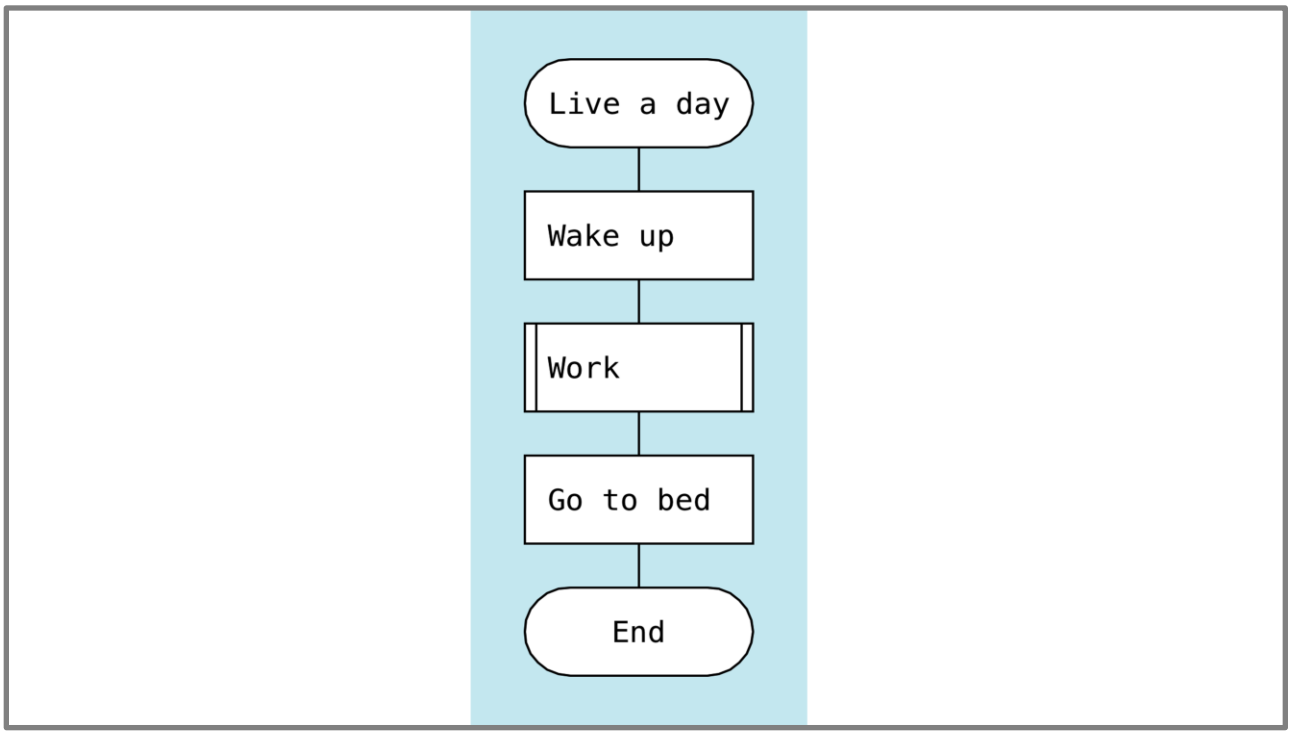
What is "metre"?



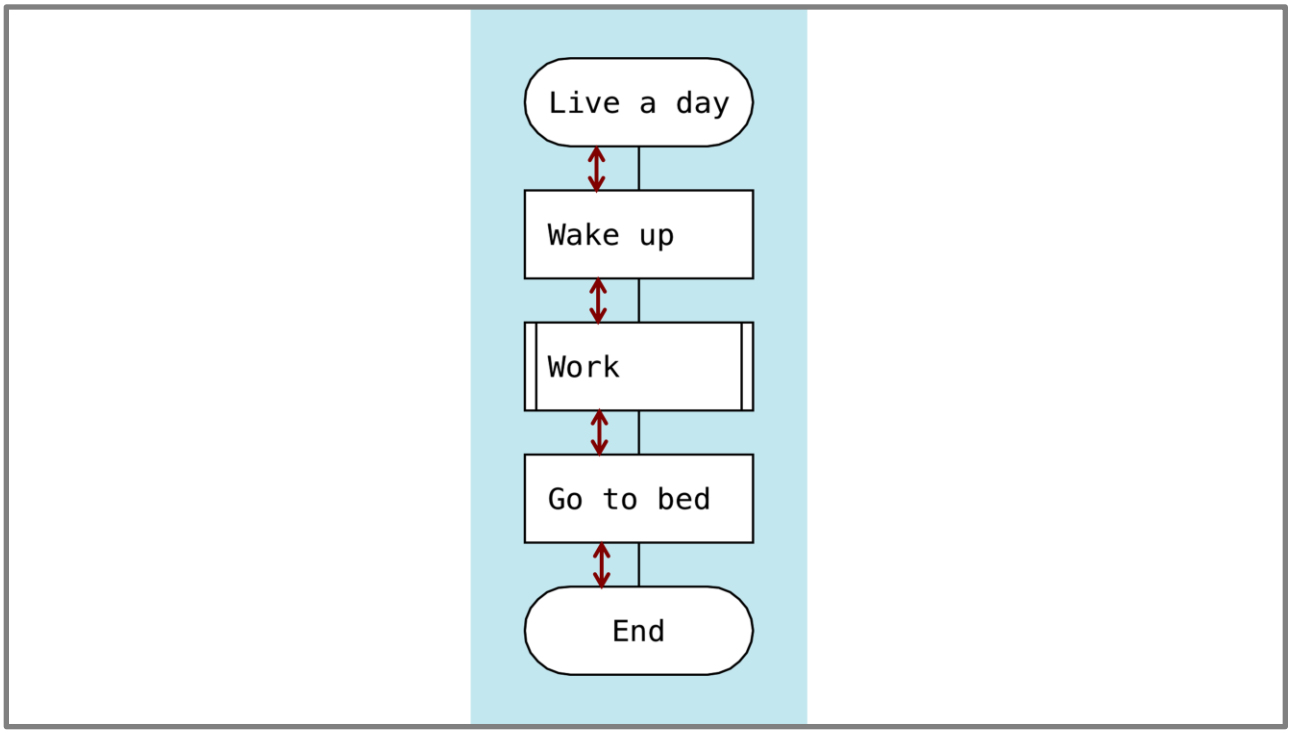
In poetry, metre is the basic rhythmic structure of a verse.



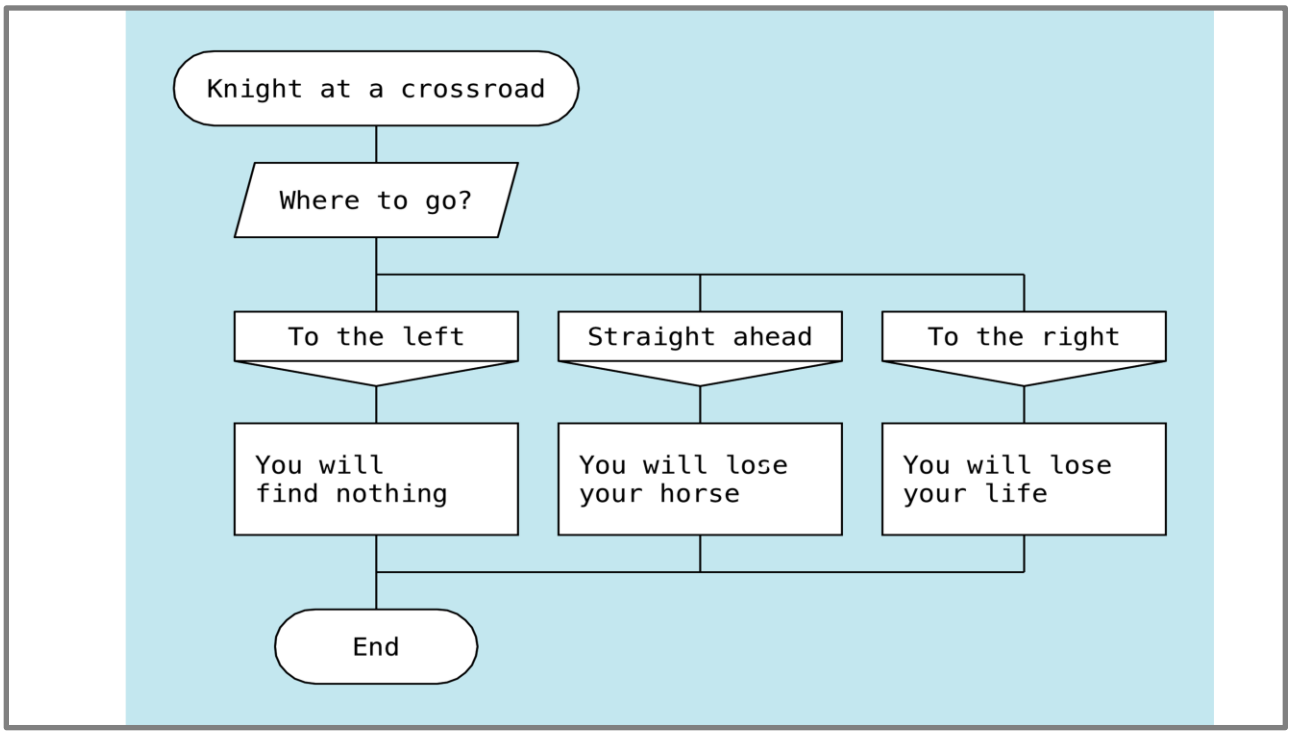
In graphics, metre is the rule that prescribes to have the same distance between any two neighbouring elements.
The viewer will not notice metre. But he or she will get an emotional feeling that the diagram looks right.



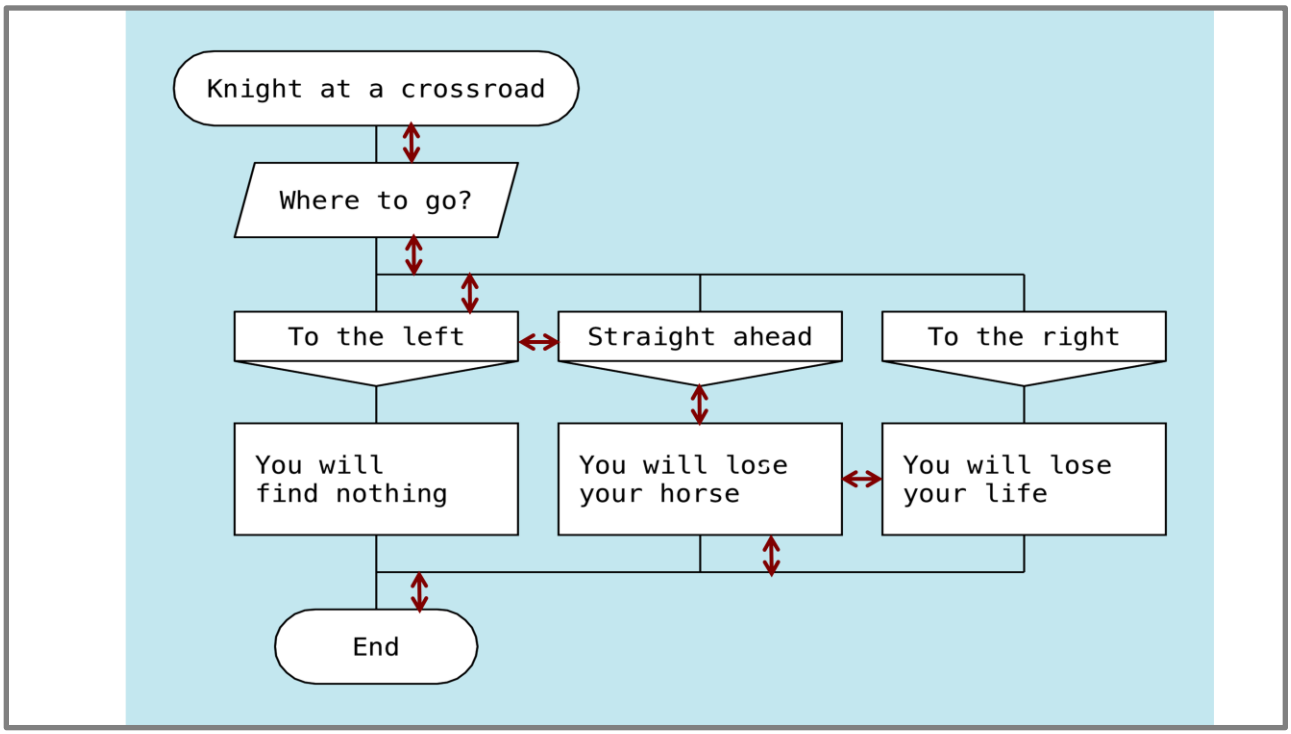
This simple diagram has metre.



These dark red arrows illustrate the same distance between icons.



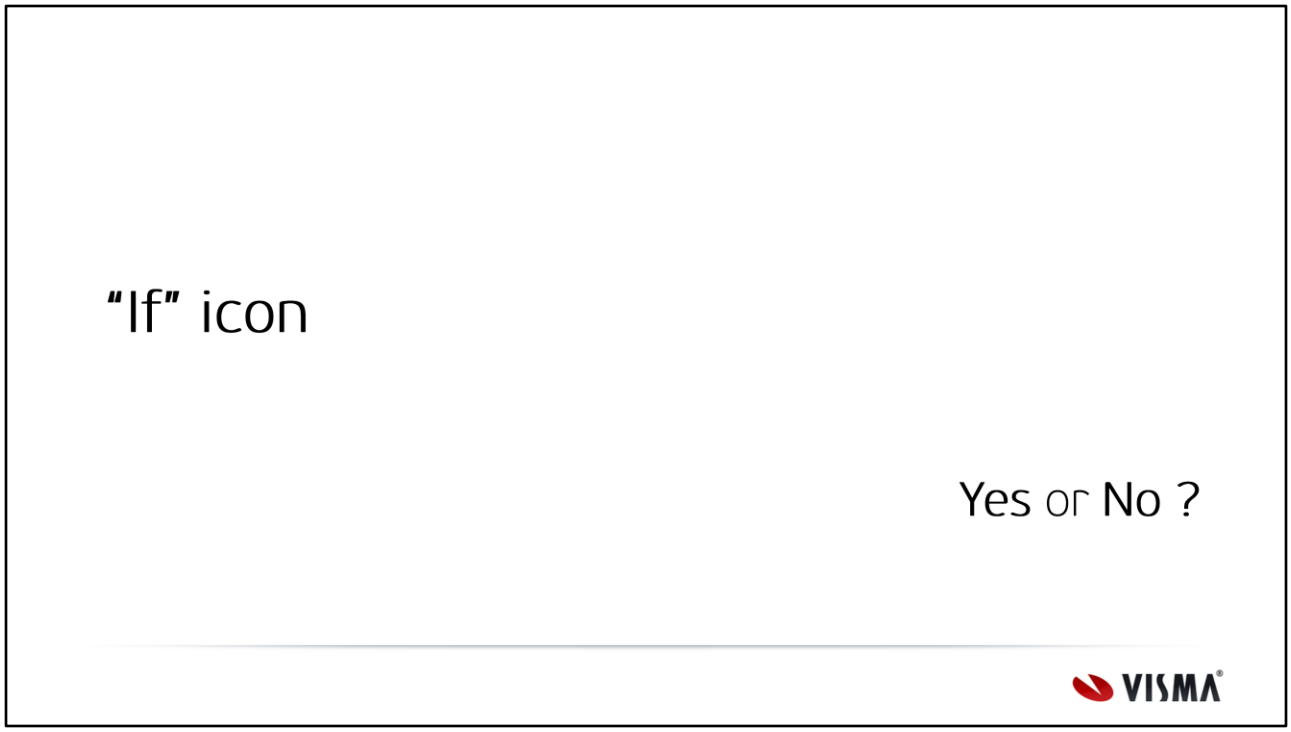
An example with several verticals. It still has metre.



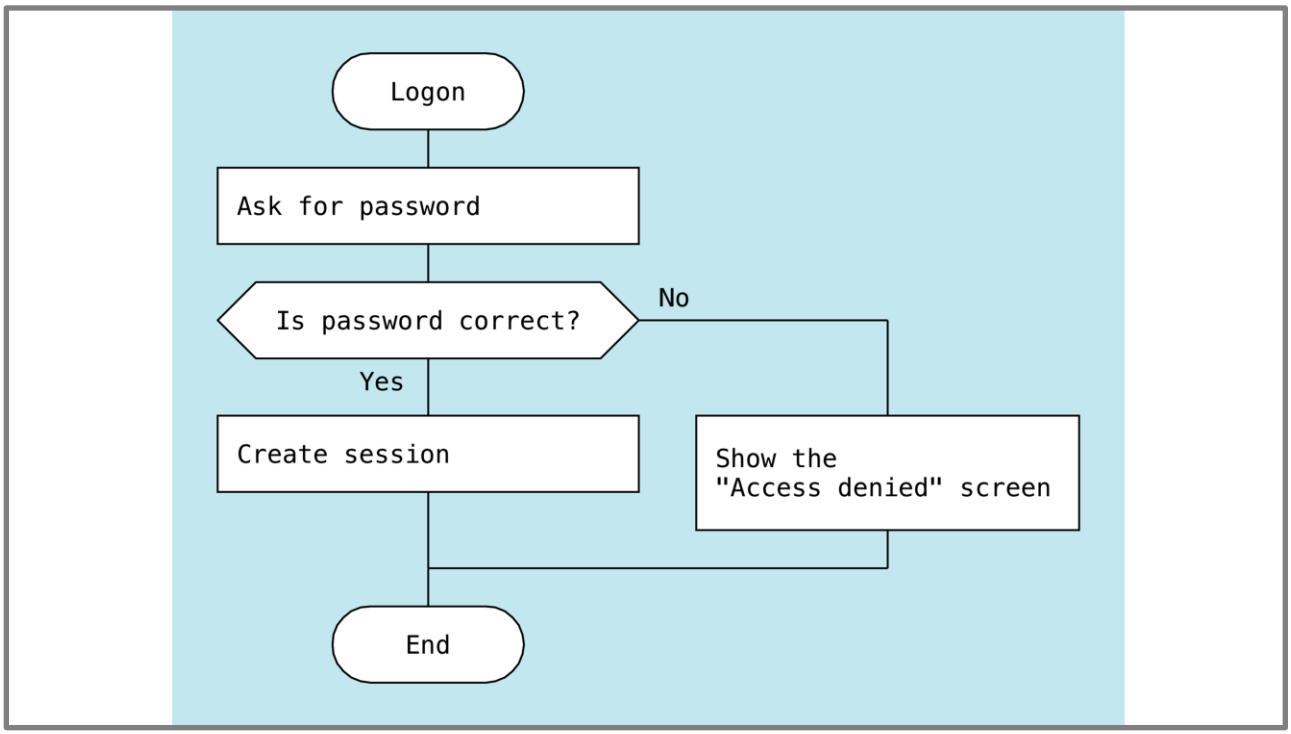
Both vertical and horizontal distances should be the same.

Decisions

Most complexity in software systems arises from decisions.
Decisions are inevitable.
DRAKON pays specific attention to showing decisions in a clear way.



The "If" icon contains a question that can be answered "yes" or "no".



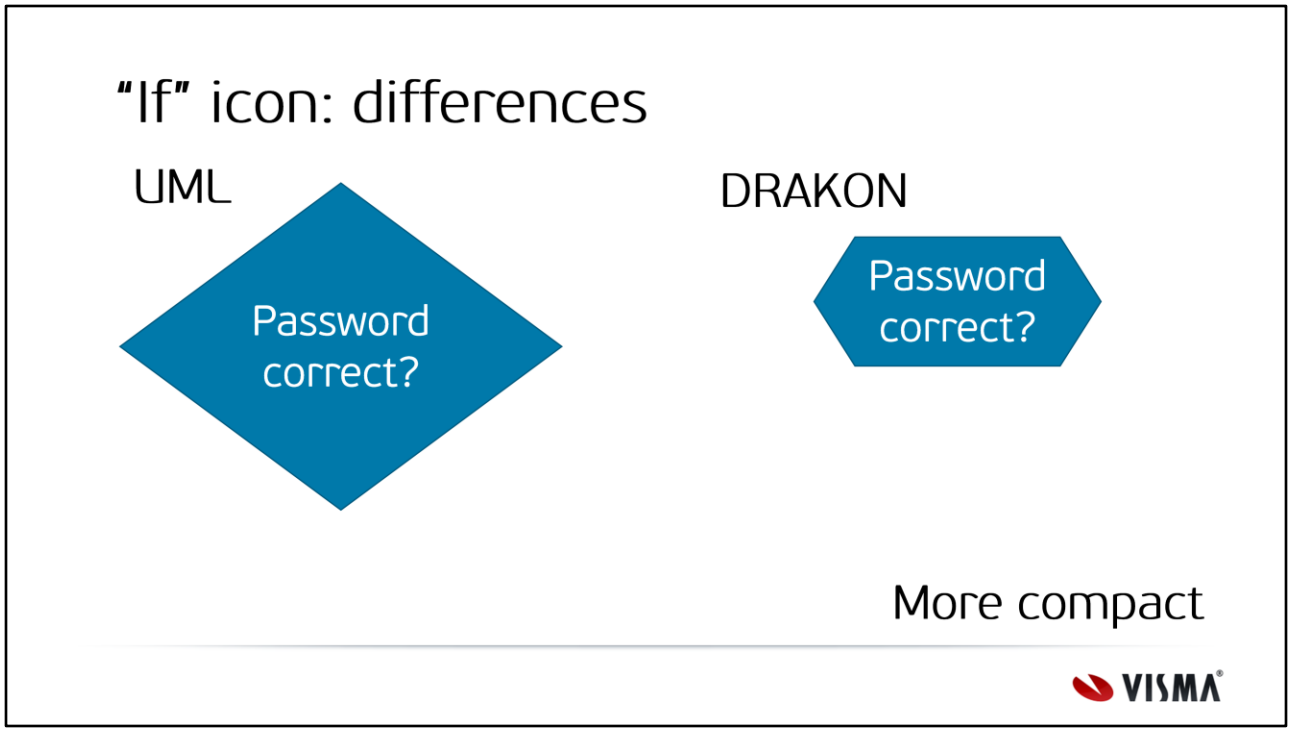
This diagram contains two use cases.

First use case: “Successful logon” (the happy path):

1. The system requests the password from the user.
2. The password is correct.
3. The systems creates a user session.

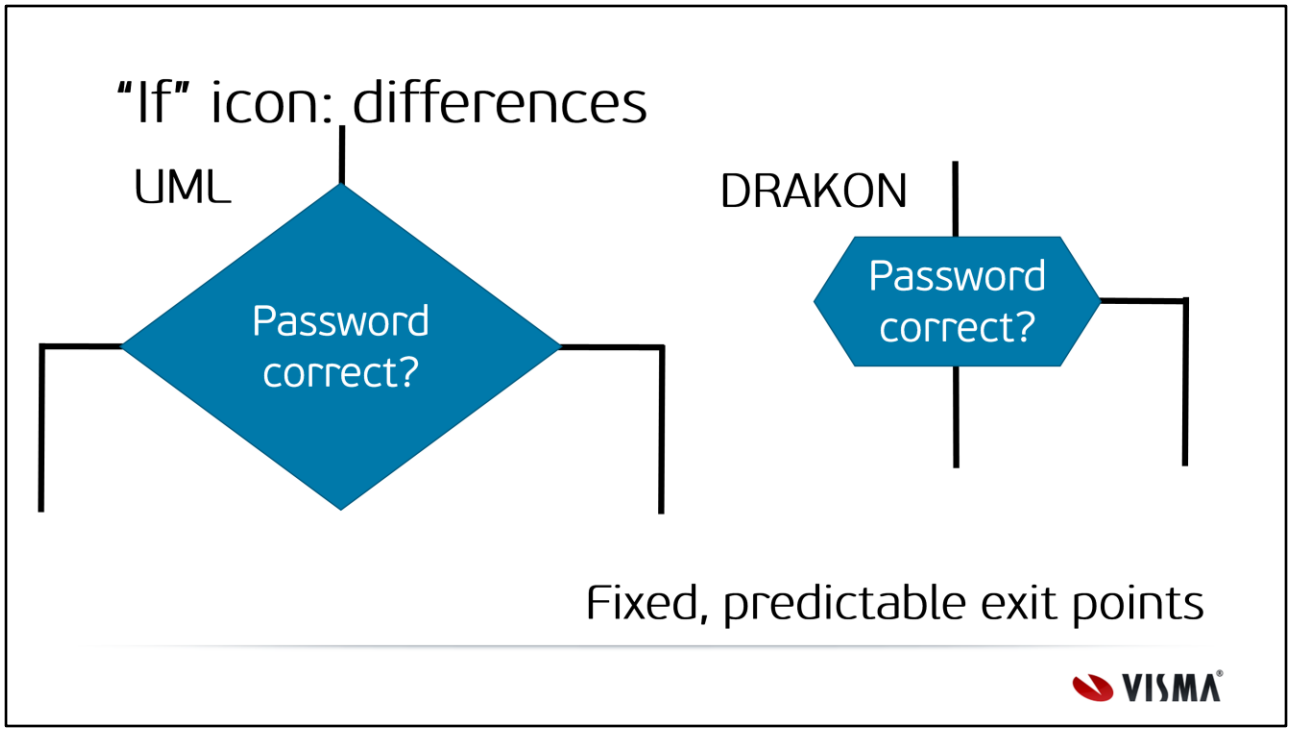
Second use case: “Access denied”

1. The system requests the password from the user.
2. The password or the user name is not correct.
3. The system shows an ‘Access denied” message.



What are the differences between the standard flow-chart "If" and DRAKON "If"?

DRAKON uses a truncated diamond instead of a diamond.
The truncated diamond takes less diagram space.



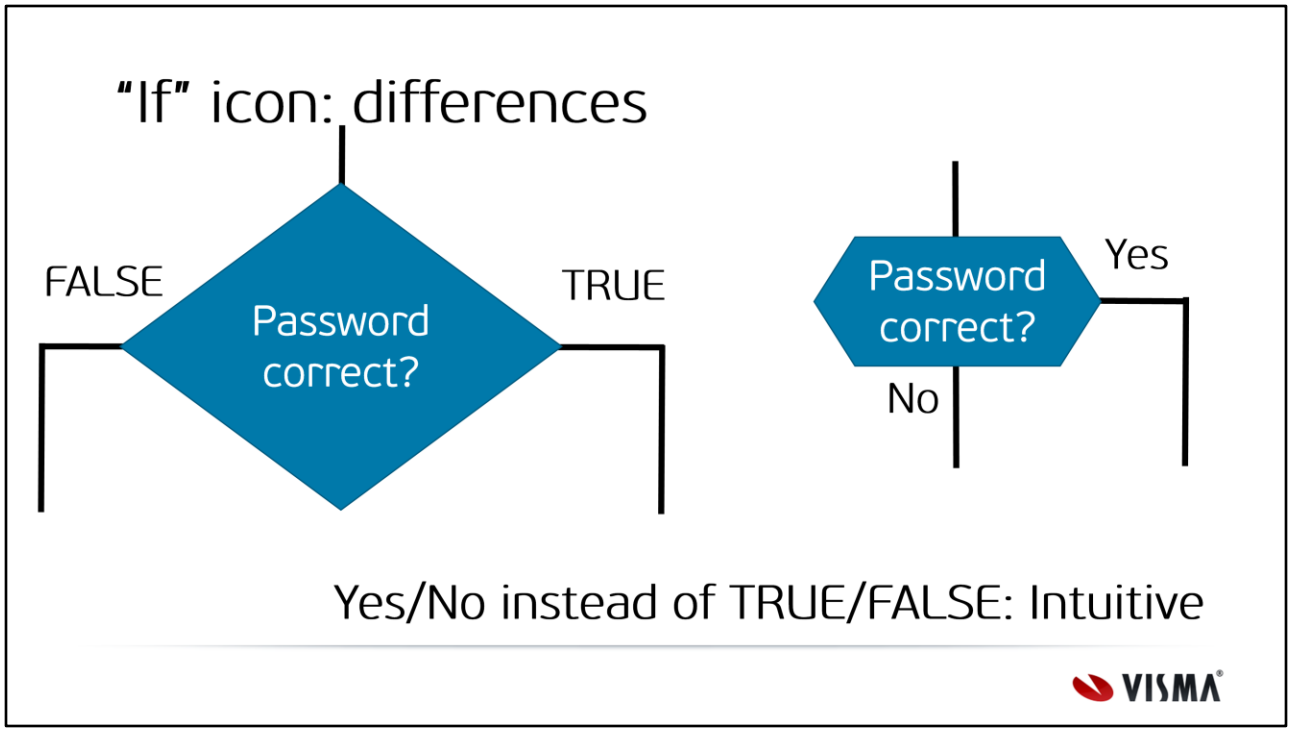
In the standard flowchart "If" icon, the entry and exit points can be placed on any corner of the diamond.

In the DRAKON "If" icon:

- The entry point is always at the top.
- The first exit point is always at the bottom.
- The second exit point is always on the right.

The "If" icon always stays on the skewer.

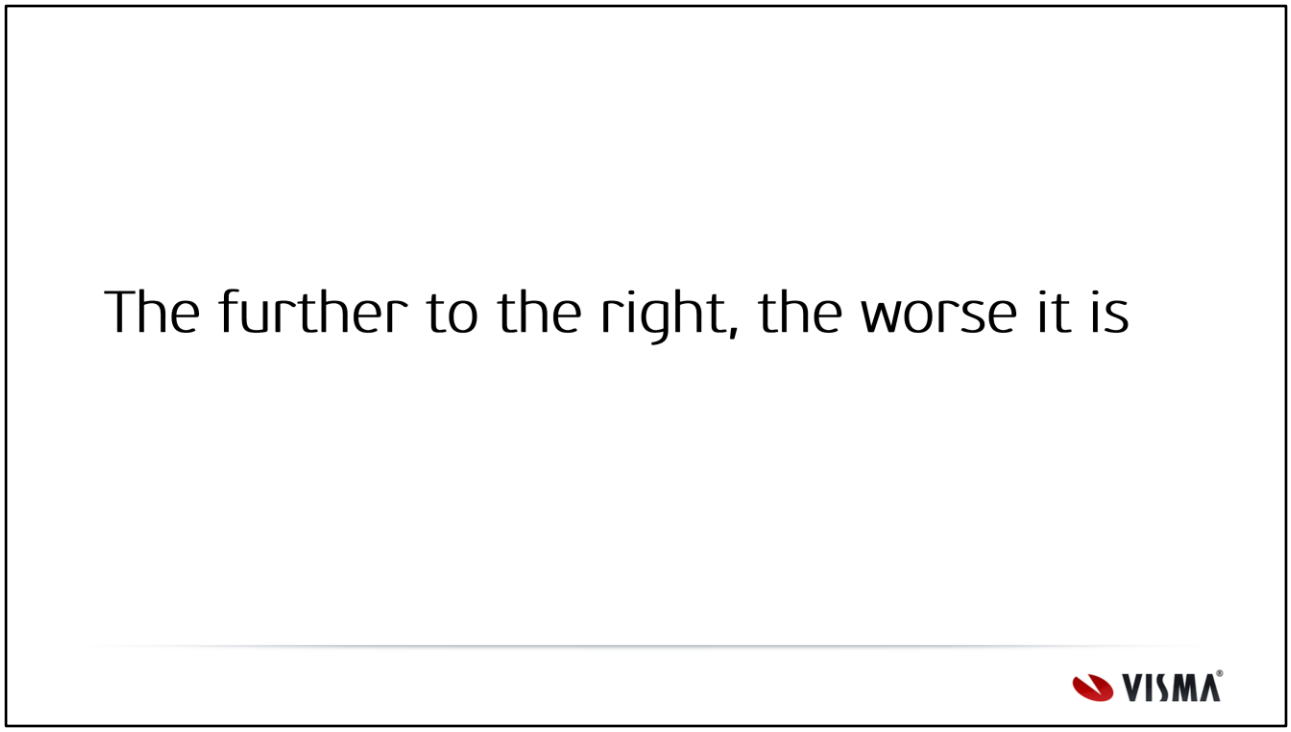
This way, the reader easily finds the exits in **expected** places.



DRAKON uses "Yes" and “No" instead of “TRUE" and “FALSE".

"TRUE" and "FALSE" look scientific and smart.

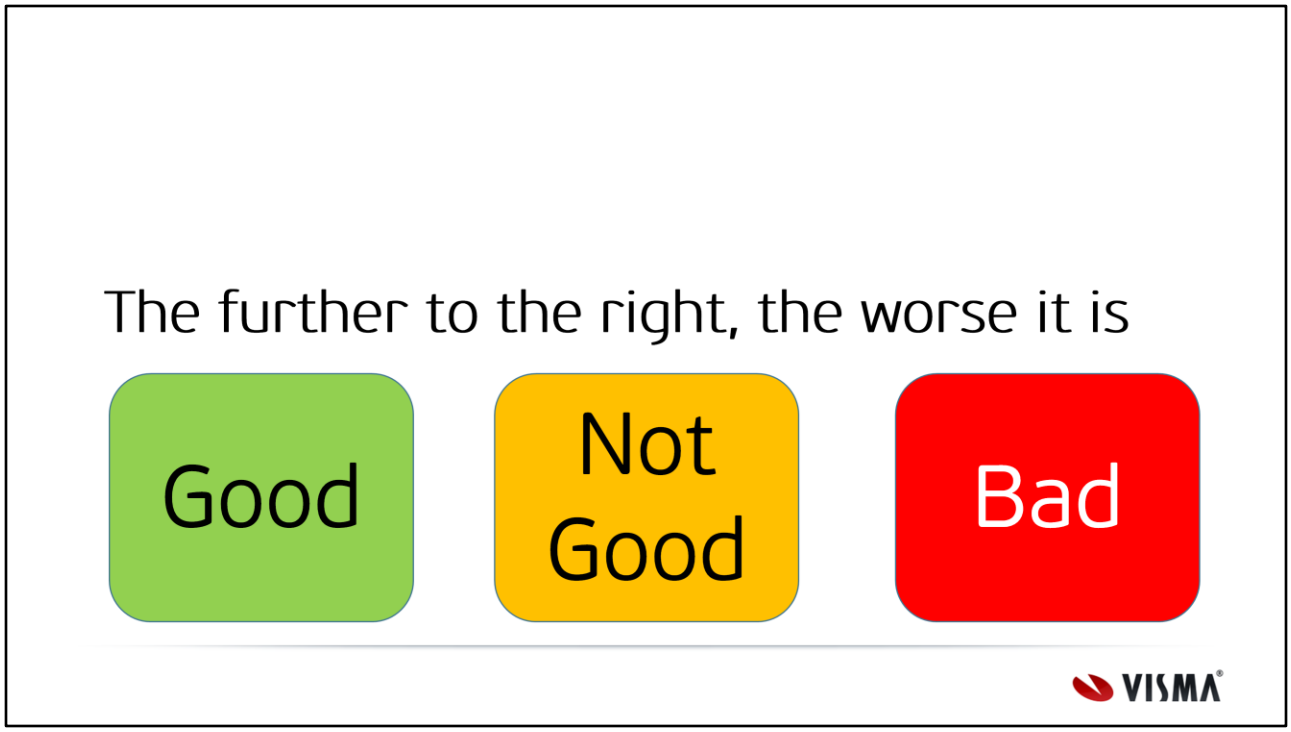
"Yes" and “No" look intuitive and clear.
"Yes" and “No" are among the first words that children learn.



The exits out of an "If" icon are not the same:

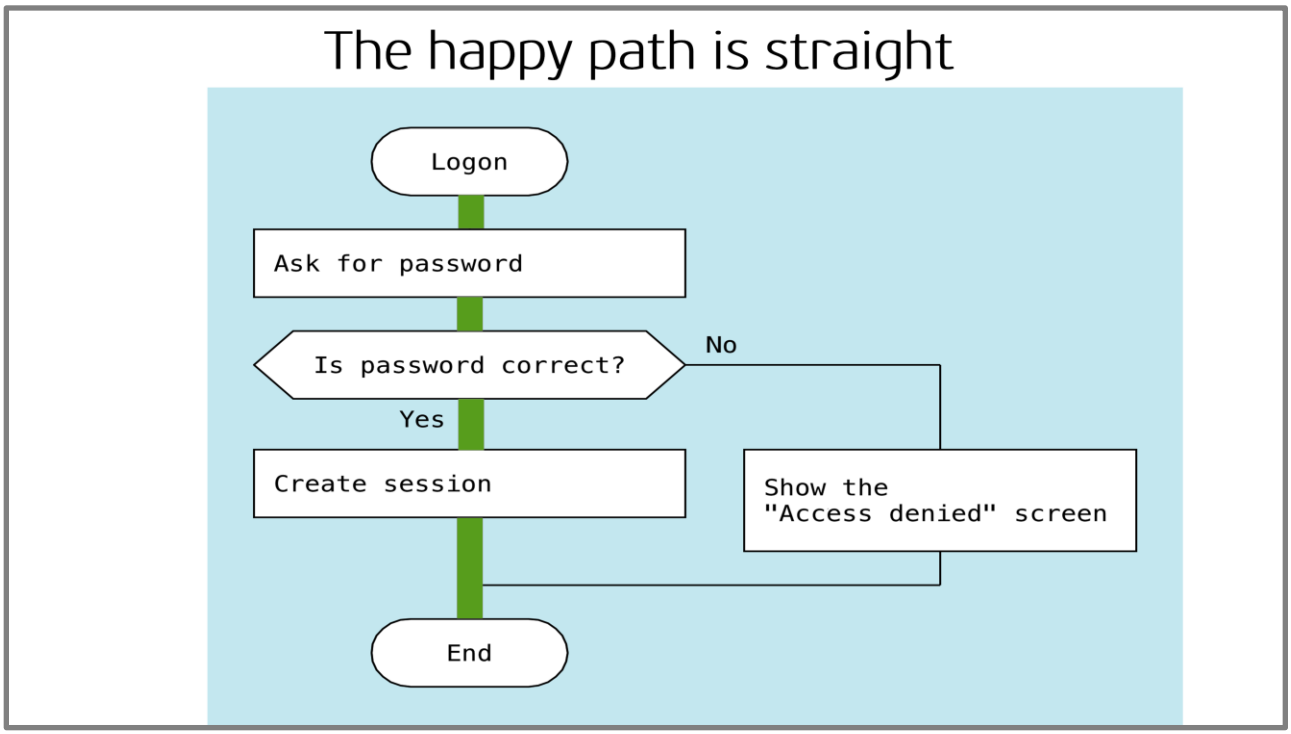
- The line that goes down is the good answer.
- The line that goes to the right is the bad answer.

The skewer holds the happy path of an algorithm.
The happy path must always be a straight vertical line.



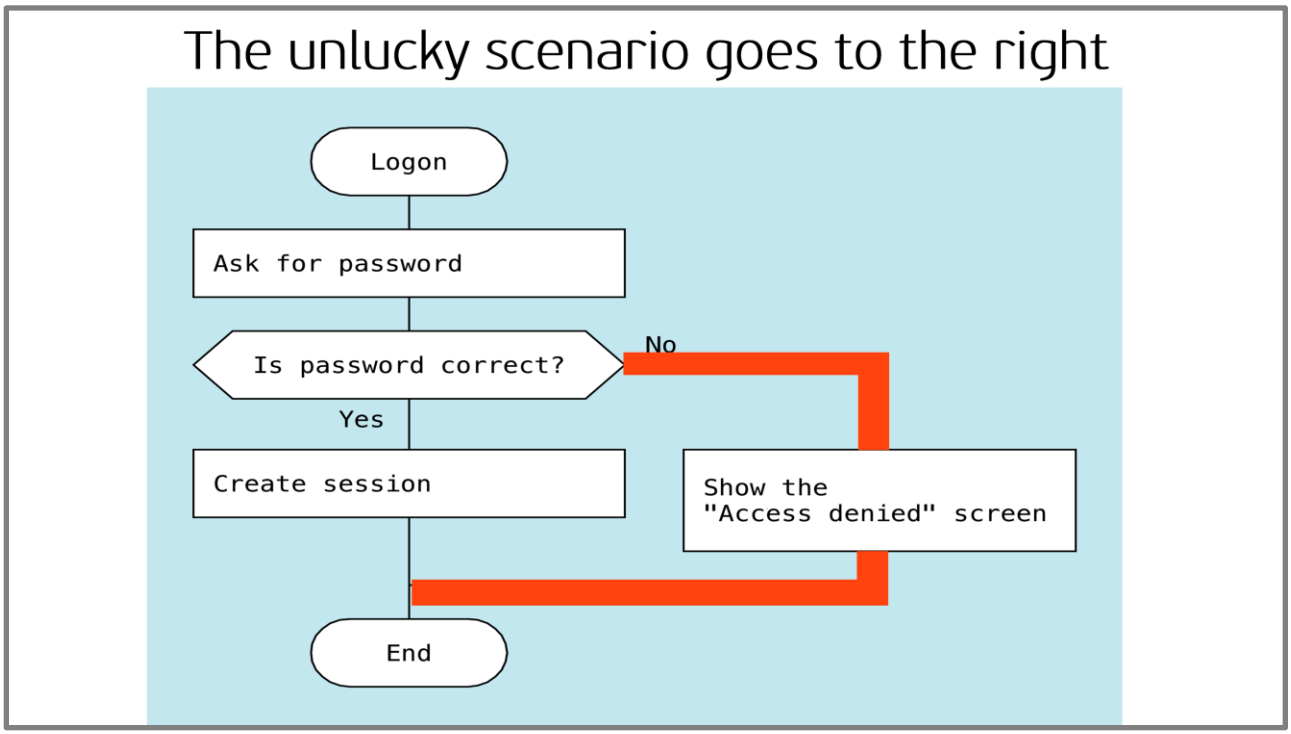
There can be several "If" icons on a diagram and several paths.
The further is a path from the skewer, the worse scenario it represents.

Often the viewer is only interested in the happy path.
In that case, he or she may skip the error-handling details.

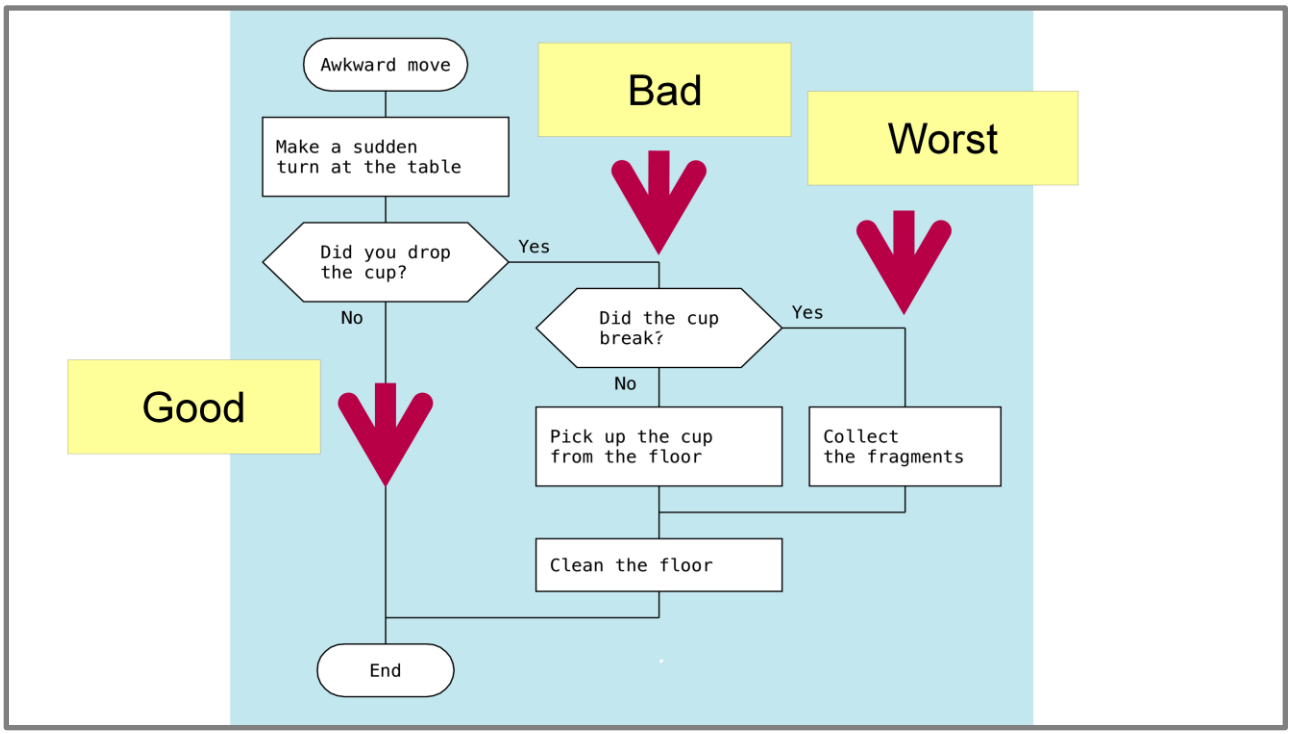


The happy path is the most successful and desirable route.

The happy path goes straight down to the end.

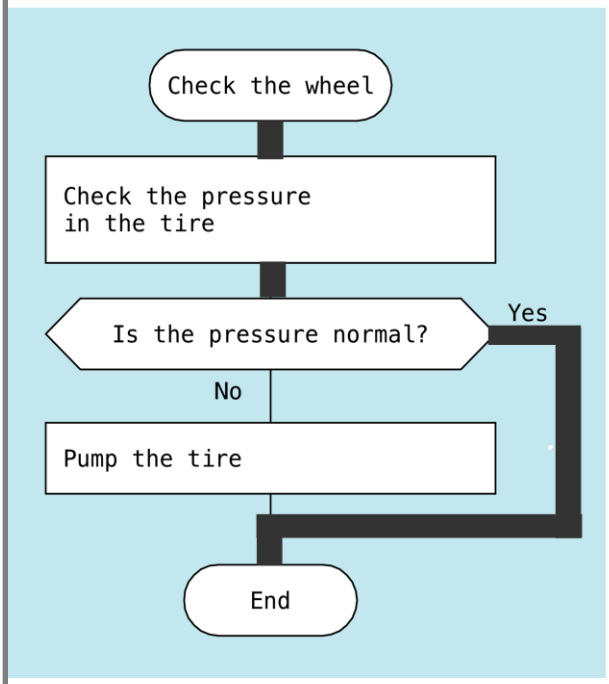


Less desirable, unwanted paths travel through the right part of the diagram.



In case of several ifs, a diagram can have many vertical lines.

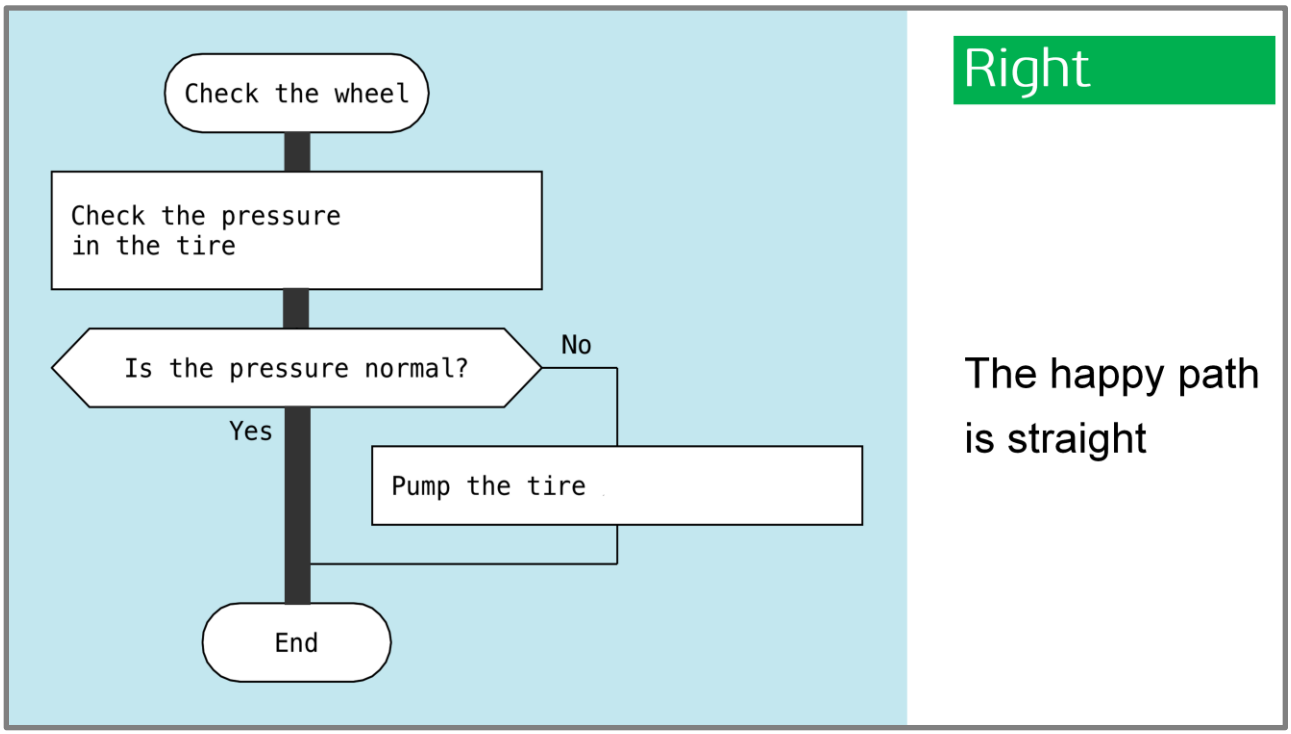
They should be arranged from left to right, from best to worst.



Wrong

The happy path is broken

This is a very serious mistake. The happy path is broken.

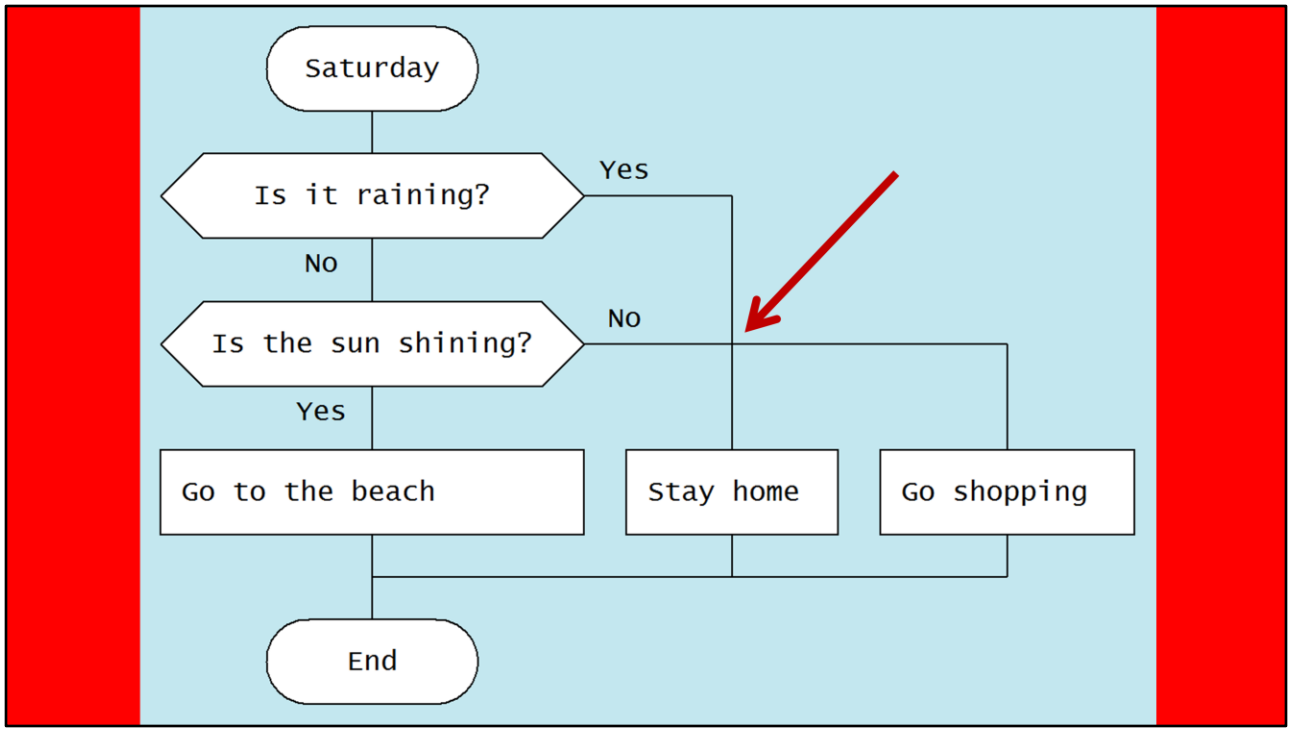


We switch the “Yes” and “No” outputs of the “If” icon and fix the error.

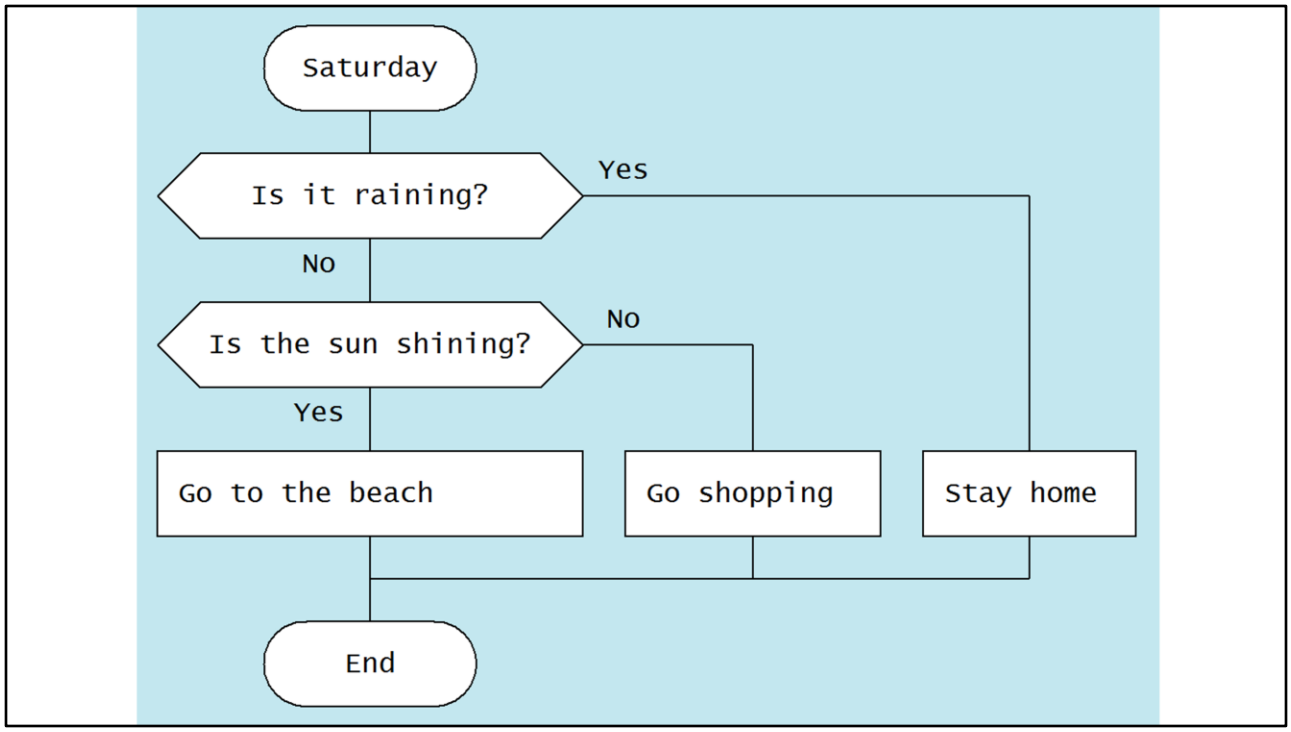
The happy path now is straight and vertical.

No intersections!

It is difficult to underestimate the importance of this rule.
Line intersections kill readability of any diagram.

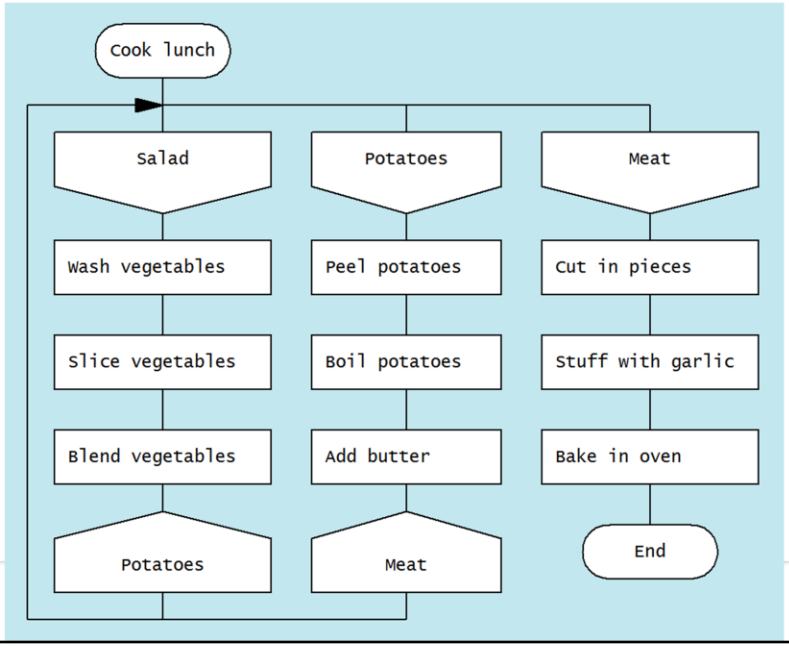


Our brain requires a lot of effort to be convinced that things that touch on the diagram are not related.
We need to free our minds from any unnecessary effort.
Let us use our energy productively. Never let lines intersect.



The intersection is removed now.

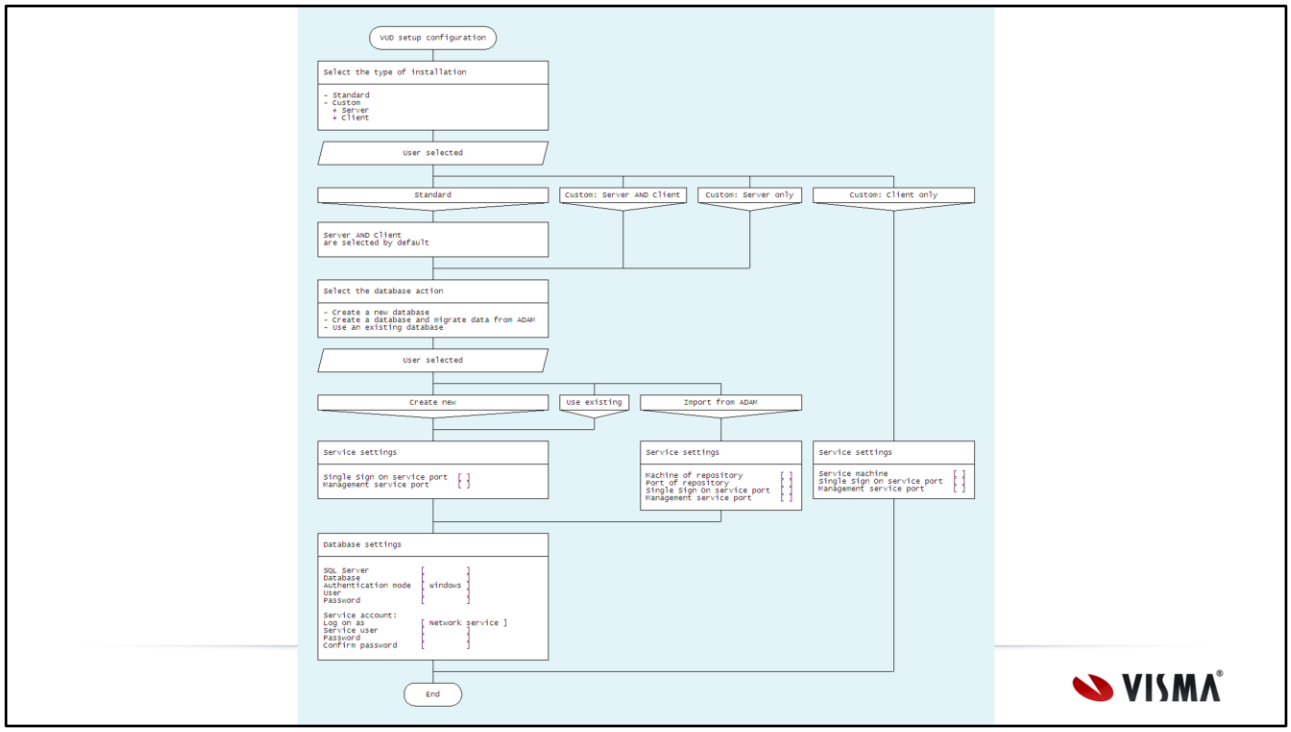
Silhouette can help



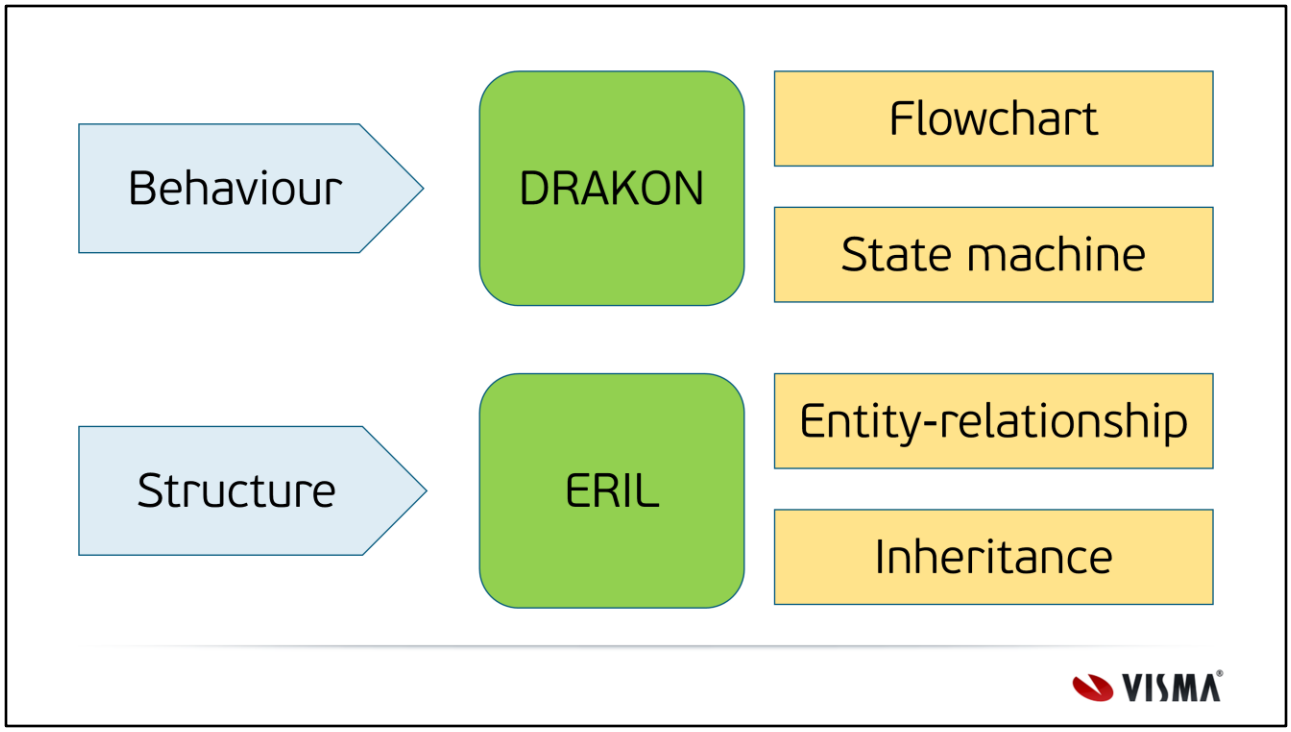
VISMA

Luckily, DRAKON allows us to avoid line intersections whatsoever.
If an algorithm is too complex, it can be split using the "silhouette" diagram type.

Silhouette divides the diagram into logical parts.



A real world example:
VUD installer.
The diagram shows which screens of the VUD installer are shown in which order according to user's input.



DRAKON represents the **behaviour** of software:

- Flowcharts
- State-machine diagrams

DRAKON diagrams have a close connection to use cases and sequence diagrams.
Each path through a DRAKON diagram is a use case.

ERIL is another visual language, similar to DRAKON.
ERIL represents **data structure**.

Thank you!

DRAKON

http://drakon-editor.sourceforge.net/language.html

ERIL

http://drakon-editor.sourceforge.net/eril.html

Stepan Mitkin stepan.mitkin@visma.com