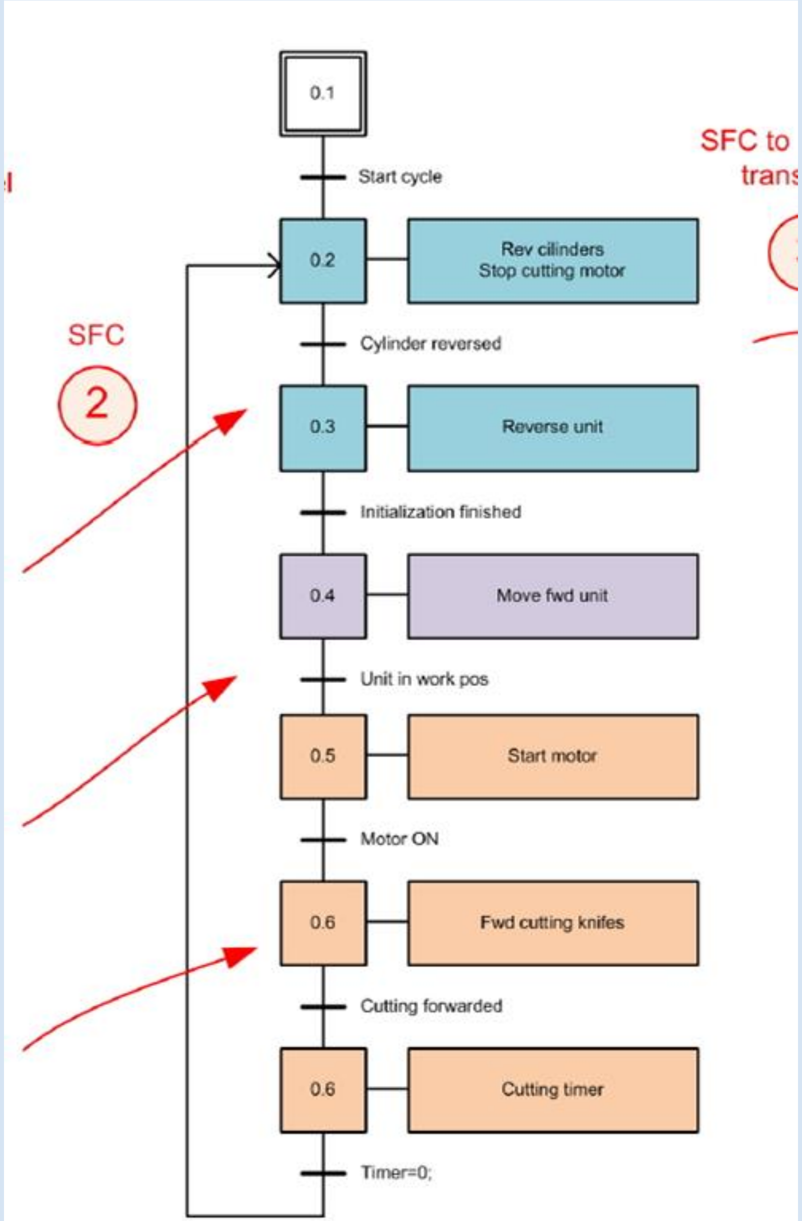



# INTRODUCTION TO SEQUENTIAL FUNCTION CHART

January 2016

**Author:** Fabrizio Avantaggiato

**Email:** avafab@gmail.com

V1.0

## Scope of this document

- Introducing the audience to Finite State Machine (FSM) modeling through a formal approach based on Sequential Function Chart (SFC), an international standard modeling language defined in IEC 61131-3.
- Giving practical examples of SFC common design patterns and best practices.
- Showing how SFC can be used to model any process in any programming languages.

# Contents




Fabrizio Avantaggiato
avafab@gmail.com

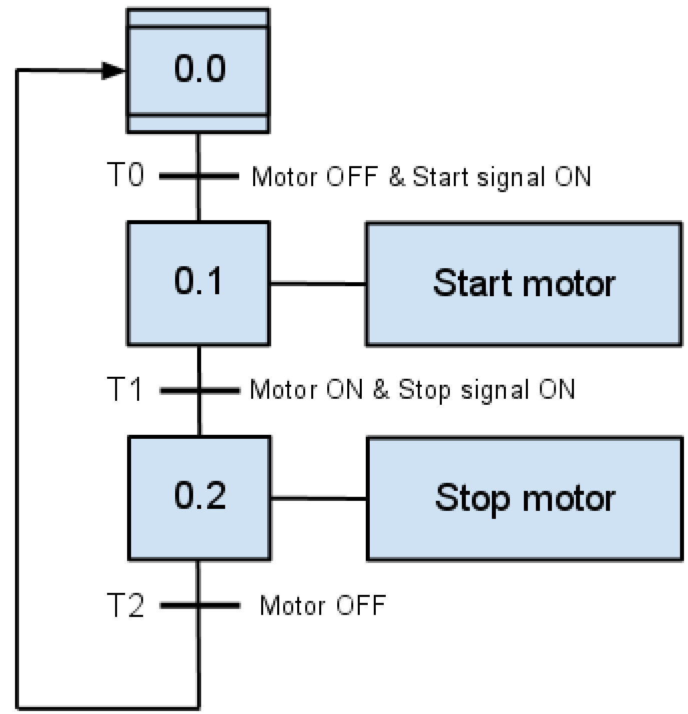
# What is SFC?

**Sequential function chart** (**SFC**) is a graphical programming language used for programmable logic controllers (PLCs).

It is one of the five languages defined by IEC 61131-3 standard.

SFC (also known as GRAFCET) is based on Petri Nets mathematical  modelling language.

It can be used to model processes that can be split into steps.

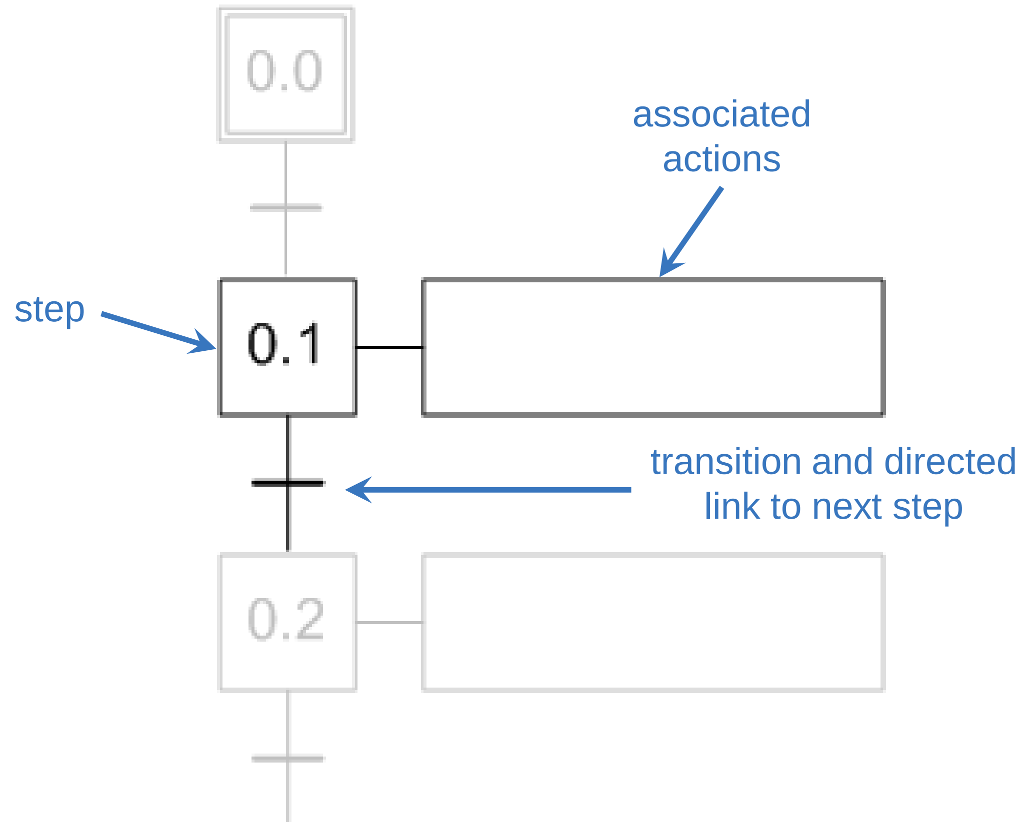



Fabrizio Avantaggiato
avafab@gmail.com

# Main components of SFC

Main components of SFC are:

- Steps with associated actions
- Transitions with associated logic conditions
- Directed links between steps and transitions




Important note: directed links connect steps and transitions with each other, no transition-transition and no step-step connection is allowed.


Fabrizio Avantaggiato
avafab@gmail.com

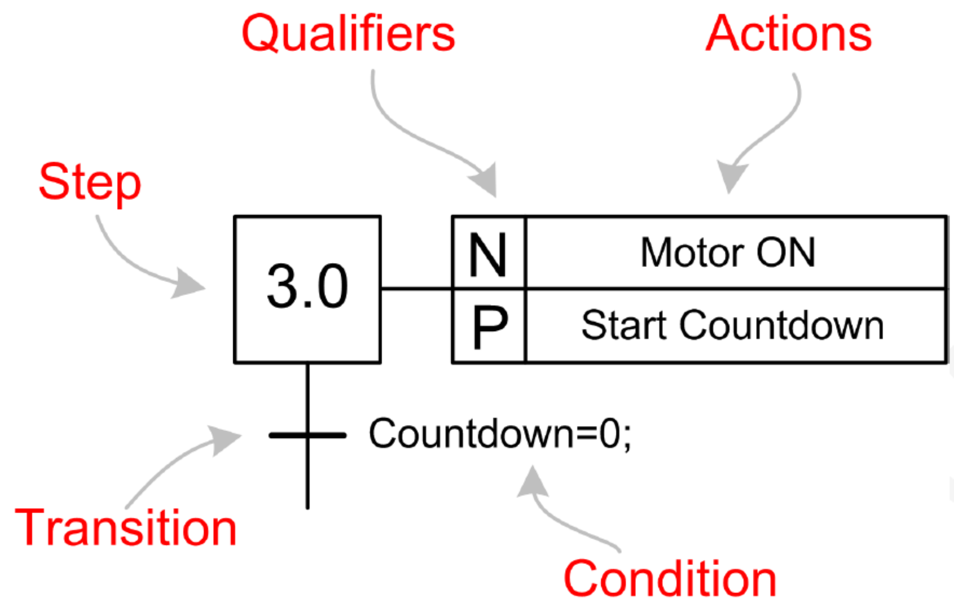
# Steps, actions and transitions

Steps in an SFC diagram can be active or inactive.

Actions are only executed if the linked step is active.

Action type is described by one of the following qualifier:

- N – non stored action
- L – time limited action
- P – pulse action
- S – stored action
- R – reset action




Fabrizio Avantaggiato
avafab@gmail.com

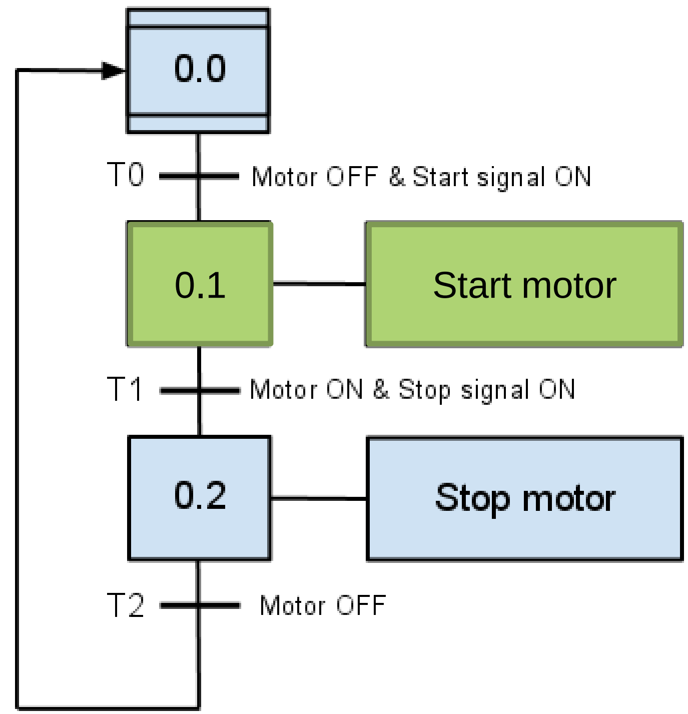
# Evolution rules

One step becomes active when both the following conditions are true:

- all steps above it are active
- All conditions on the connecting transition are true

When a transition is fired,

- all steps above are deactivated
- all steps below are activated




Fabrizio Avantaggiato
avafab@gmail.com

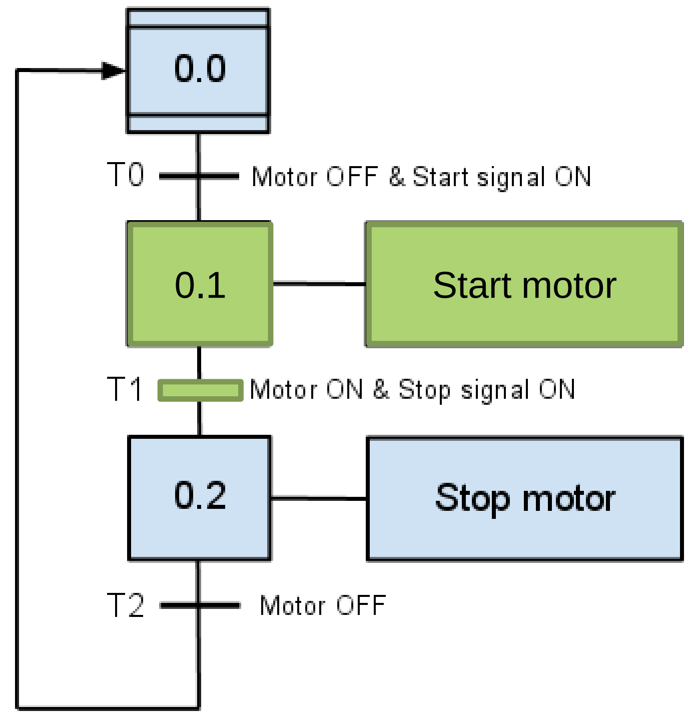
# Evolution rules

One step becomes active when both the following conditions are true:

- all steps above it are active
- All conditions on the connecting transition are true

When a transition is fired,

- all steps above are deactivated
- all steps below are activated




Fabrizio Avantaggiato
avafab@gmail.com

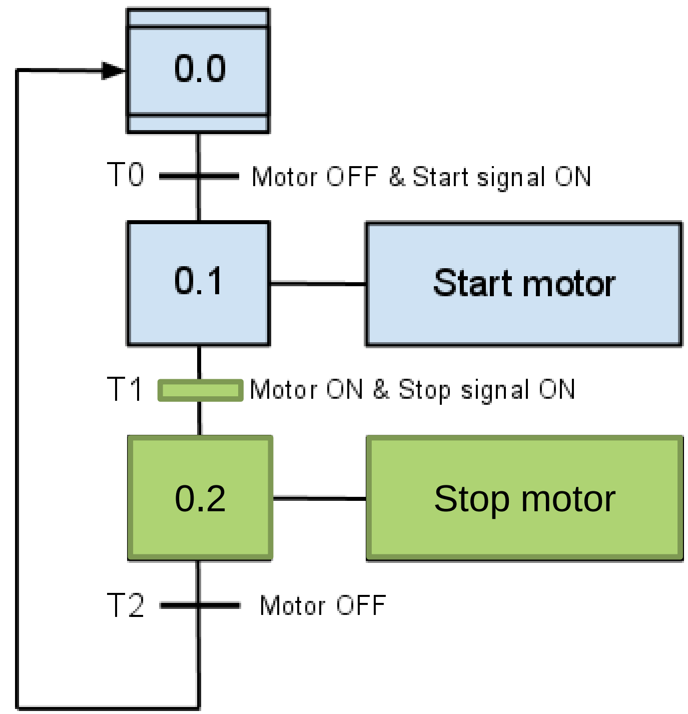
# Evolution rules

One step becomes active when both the following conditions are true:

- all steps above it are active
- All conditions on the connecting transition are true

When a transition is fired,

- all steps above are deactivated
- all steps below are activated





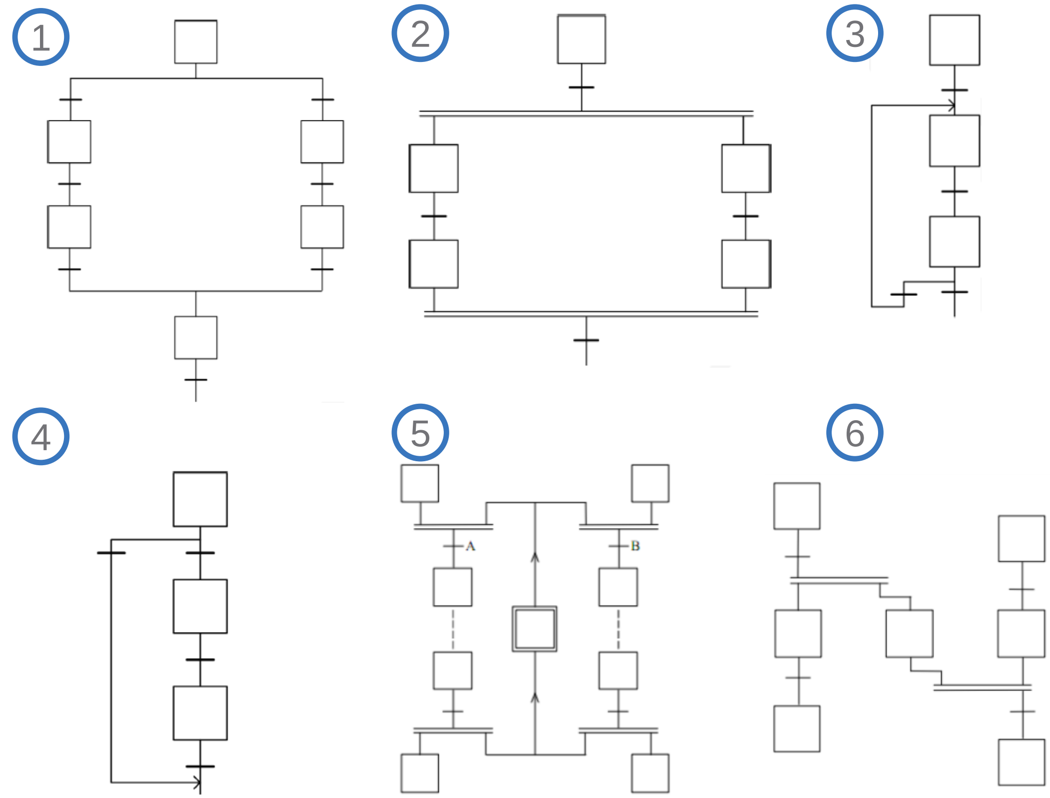
Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



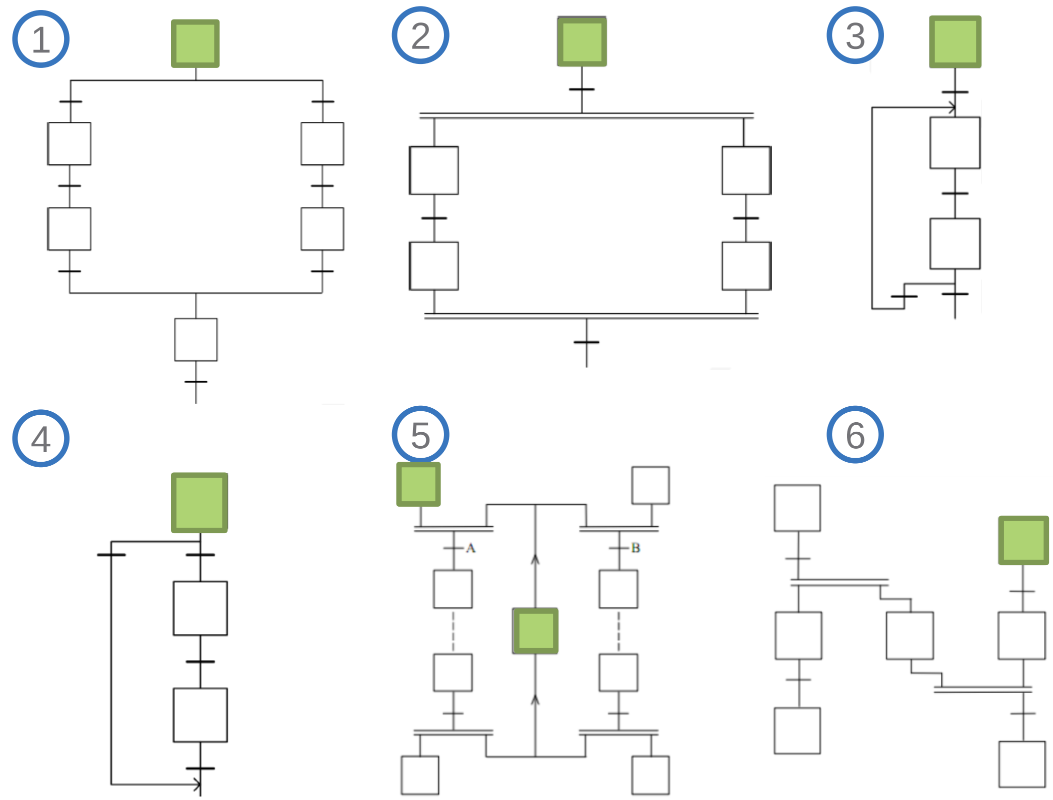

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



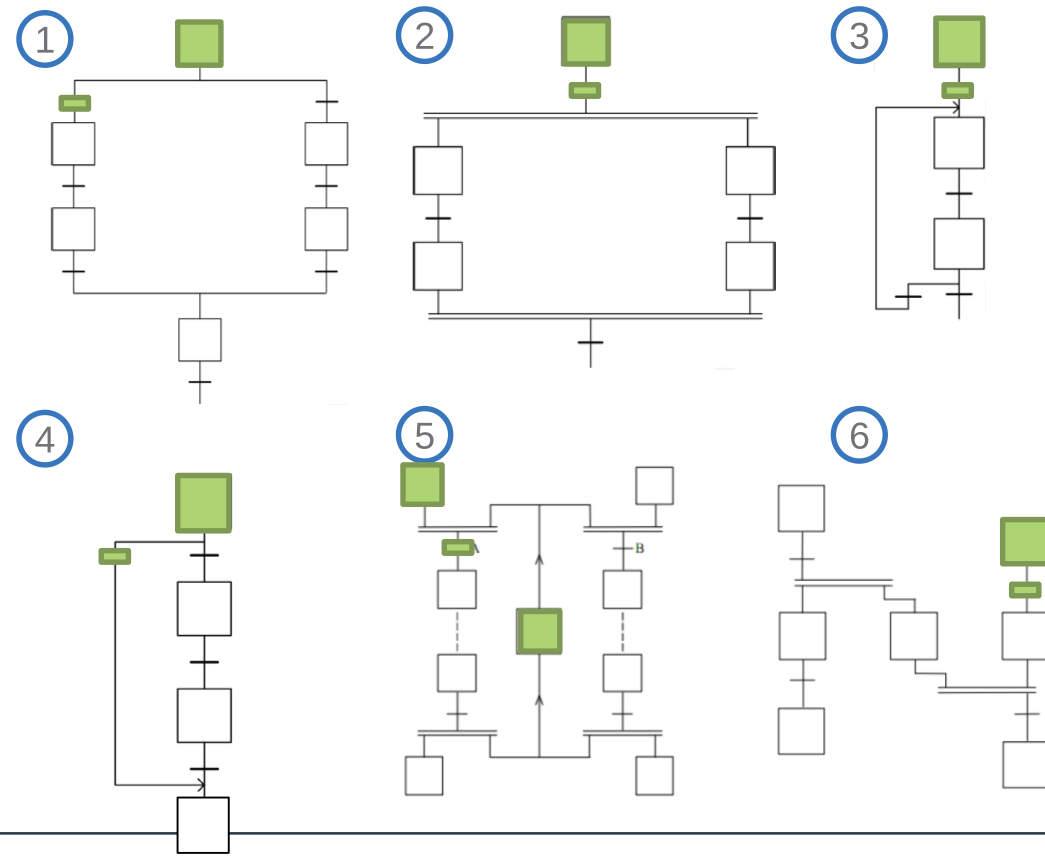

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization





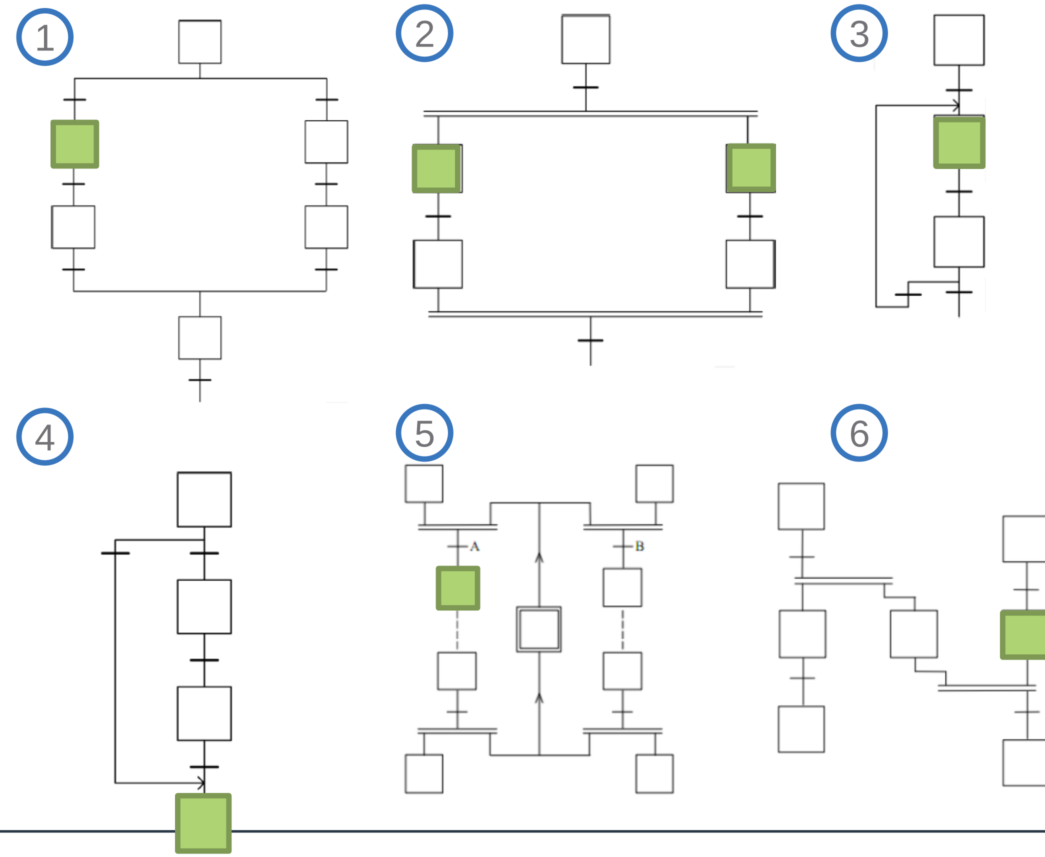
Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



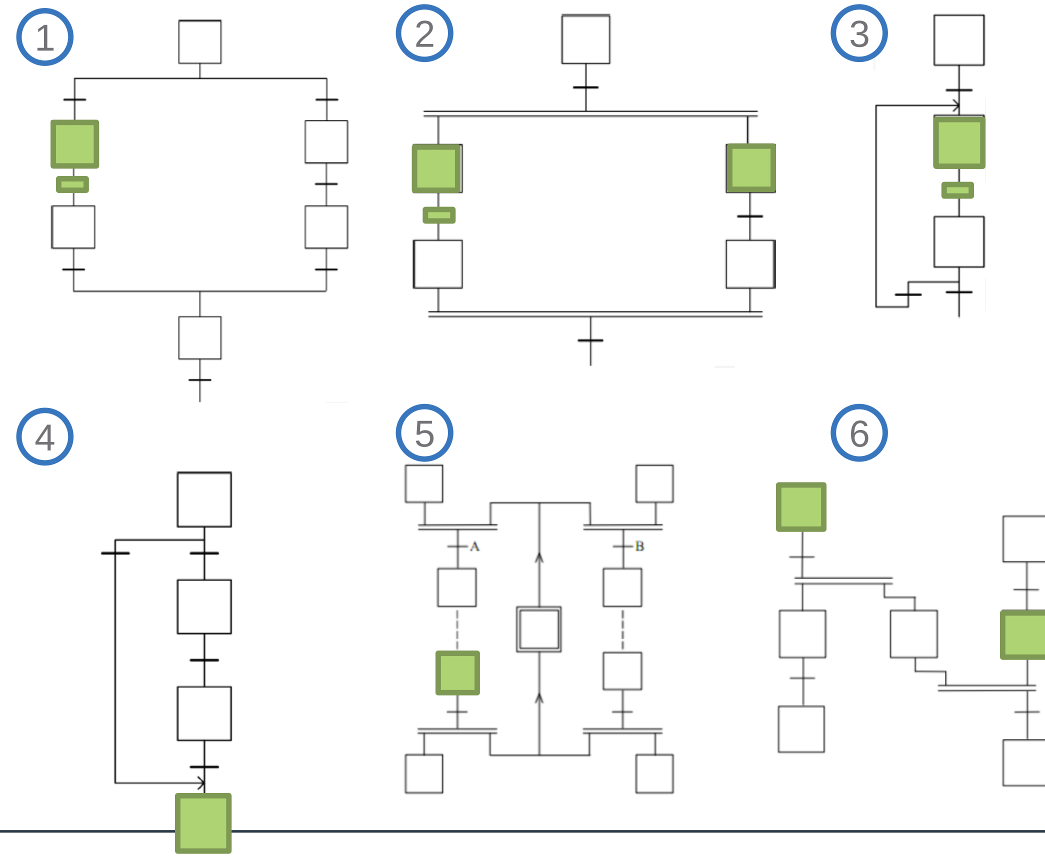

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



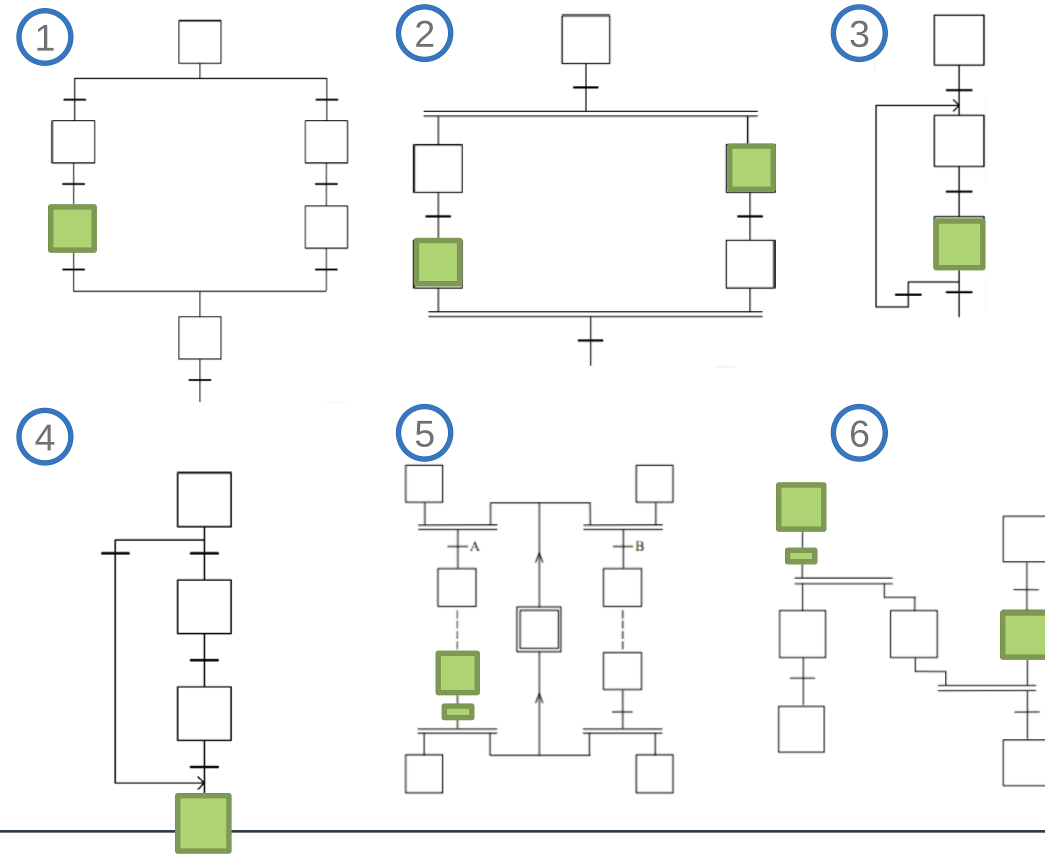

Fabrizio Avantaggiato
avafab@gmail.com

## Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



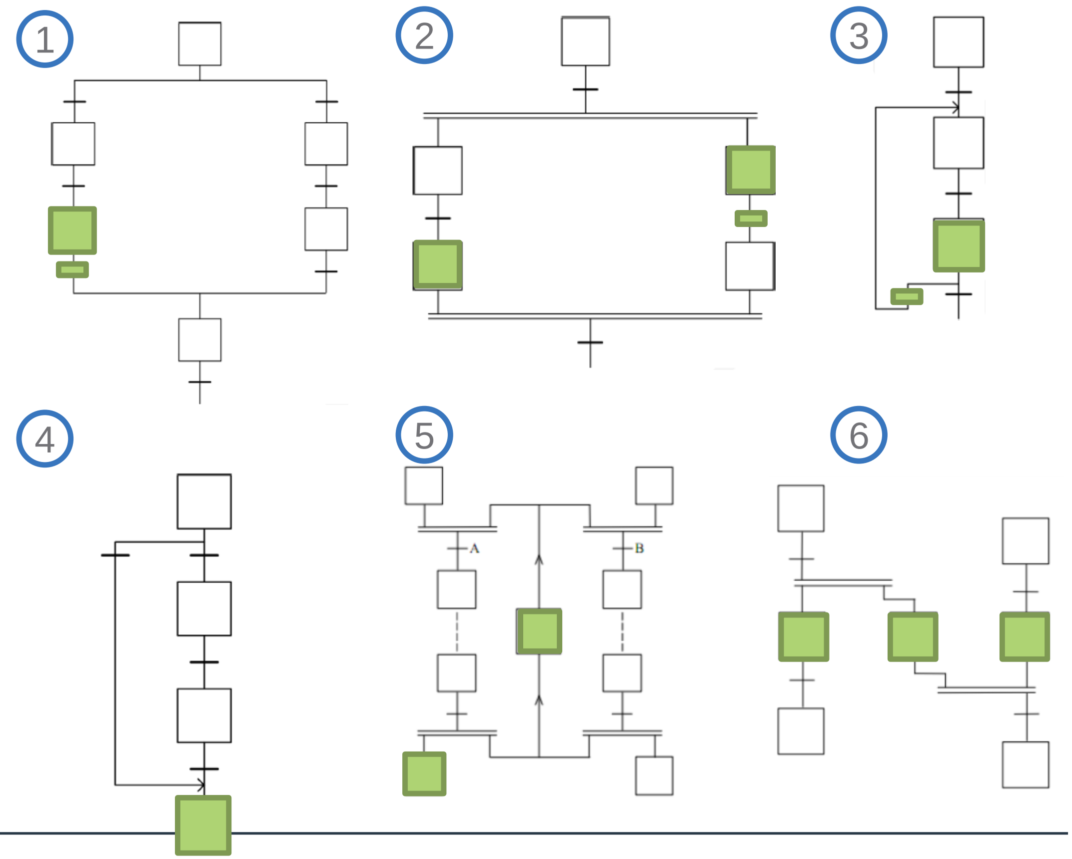

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



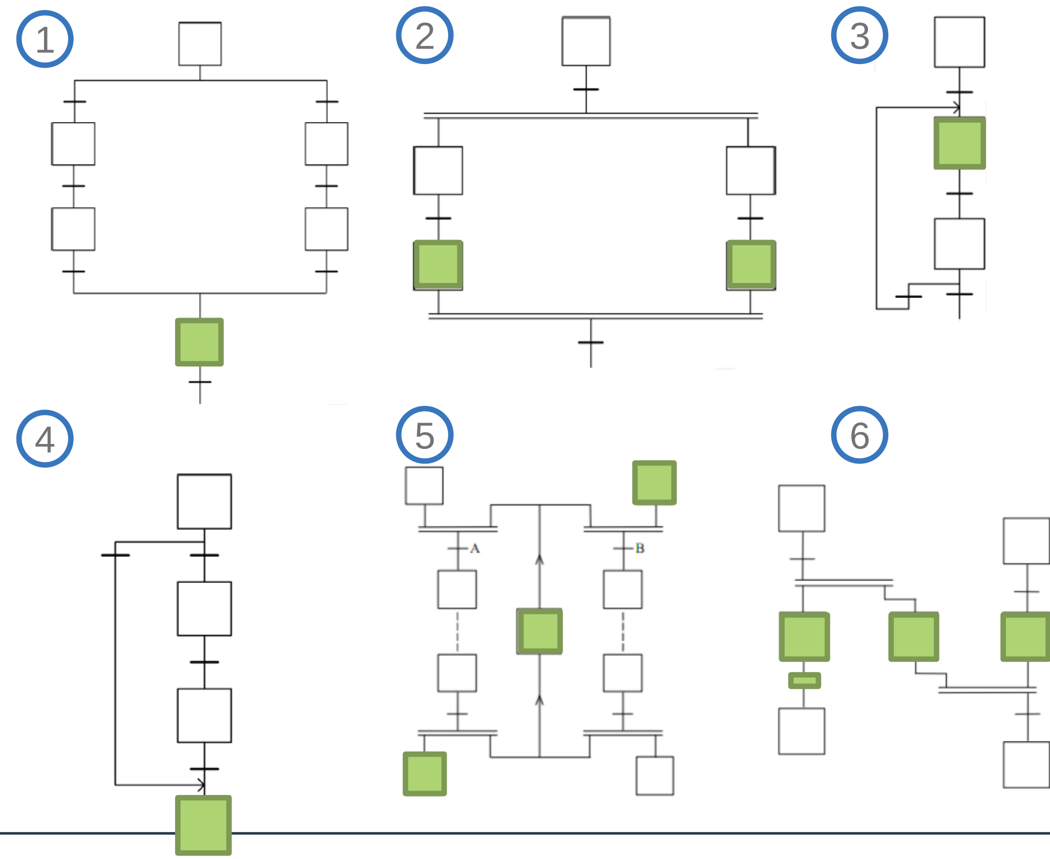

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



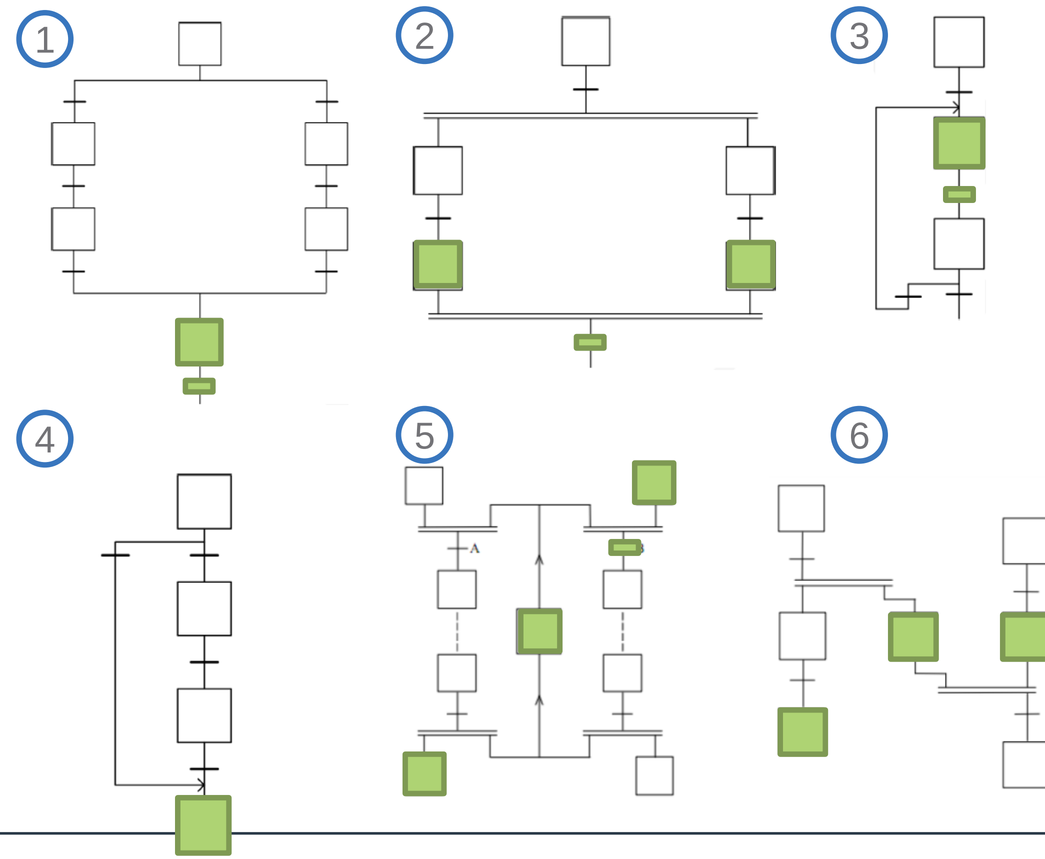

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



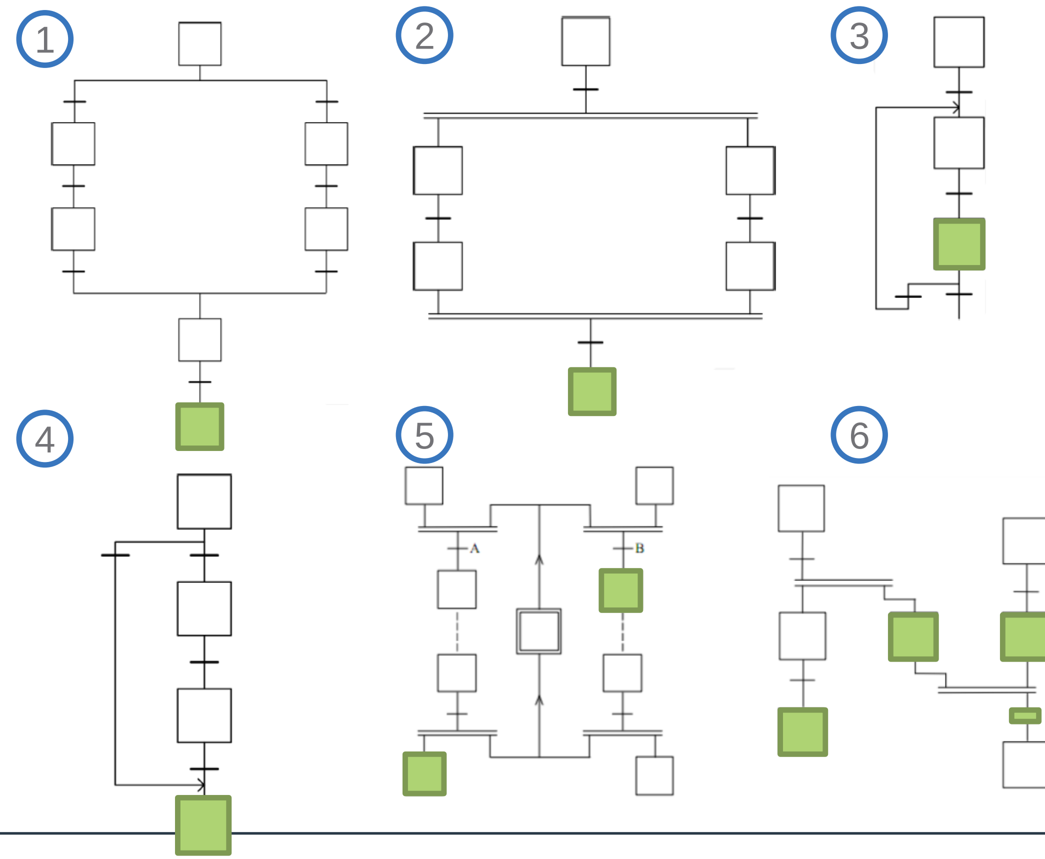

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



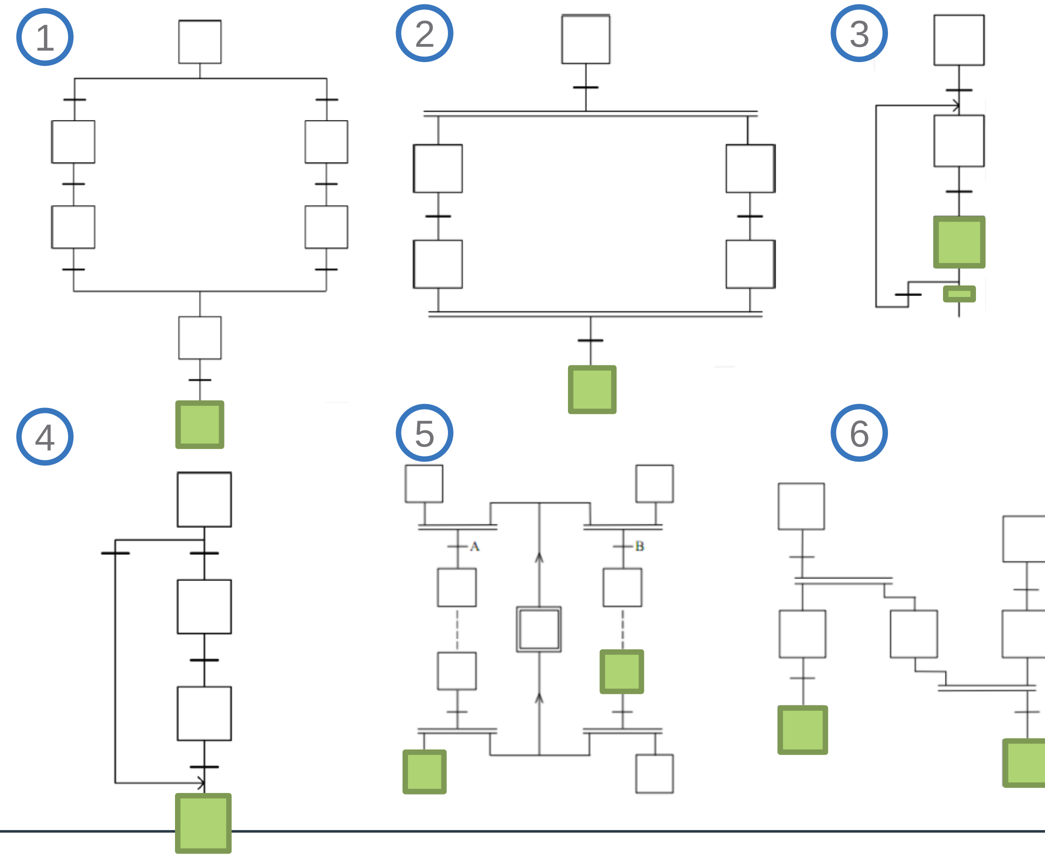

Fabrizio Avantaggiato
avafab@gmail.com

## Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization





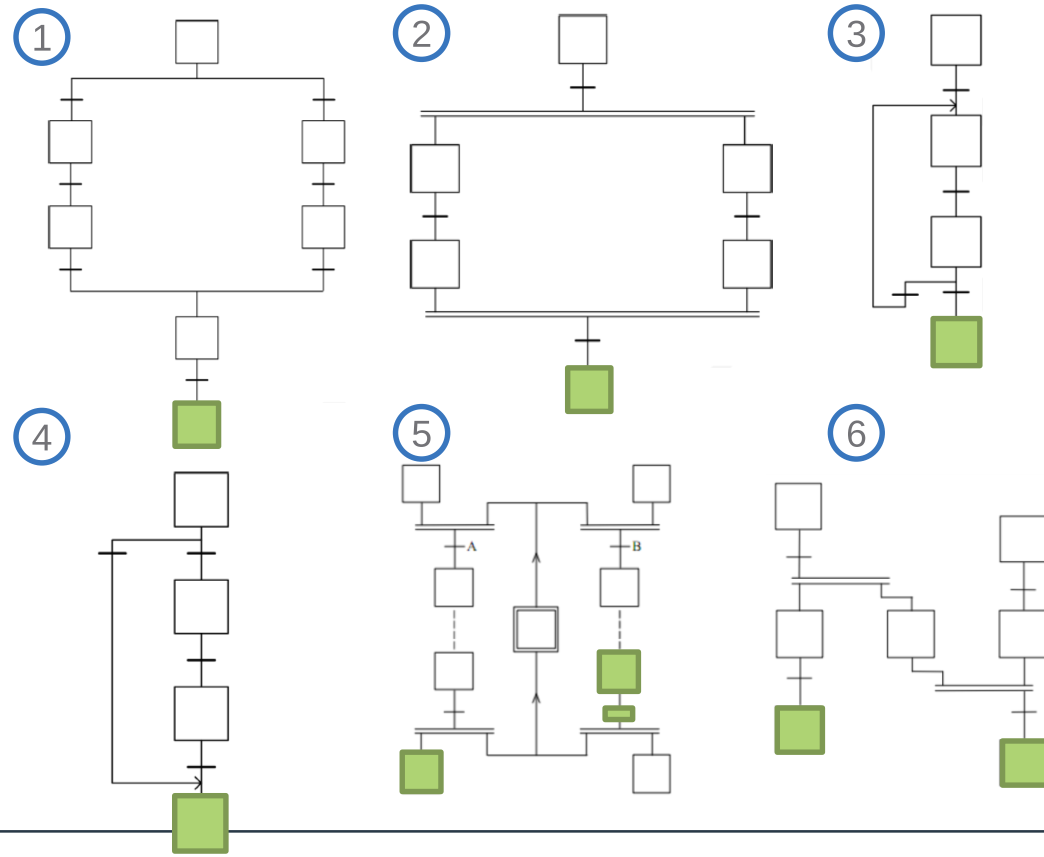
Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



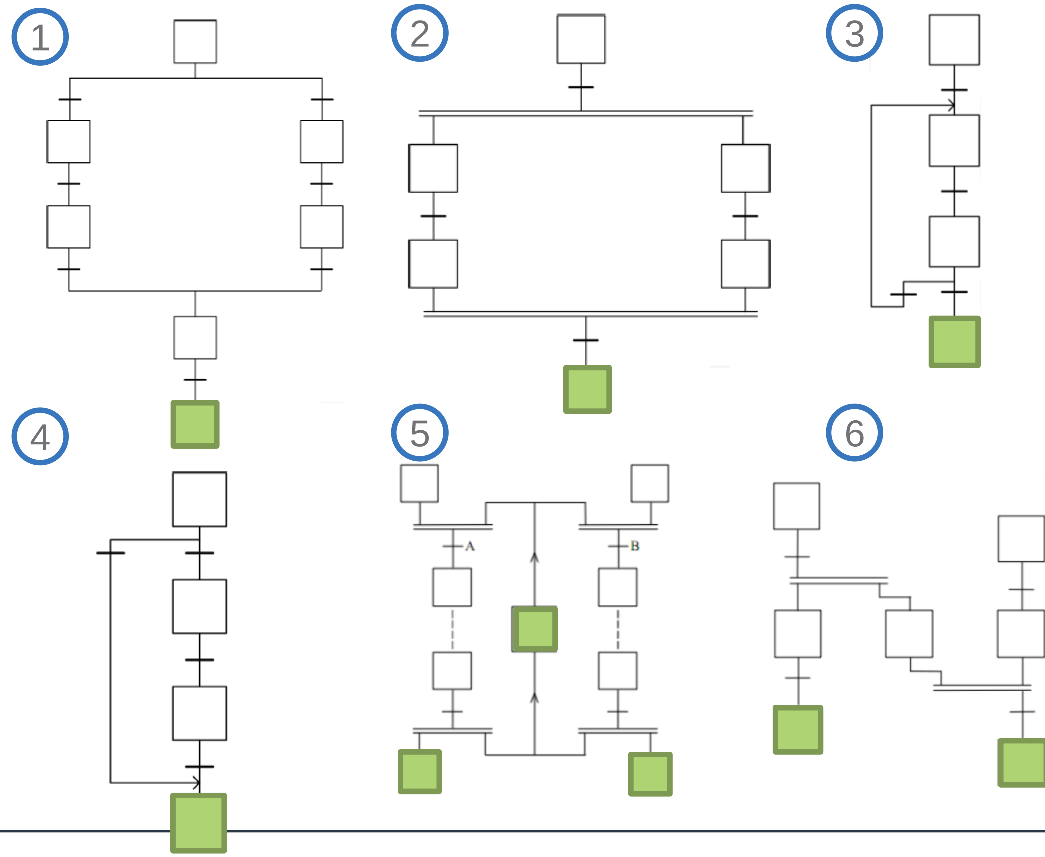

Fabrizio Avantaggiato
avafab@gmail.com

# Design structures

Every finite state machine can be modeled mixing together steps, actions and transitions following design and evolution rules.

There are 6 design structures to model common finite state machine behaviors:

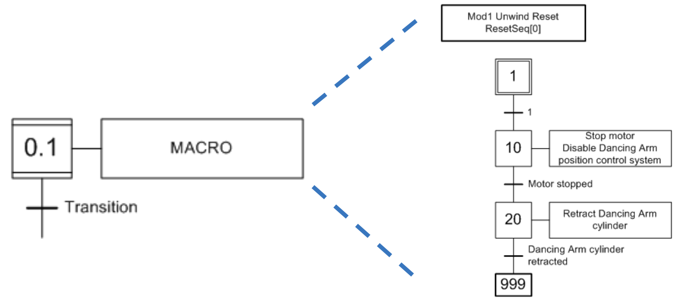
1. Choice and convergence
2. Parallelism and synchronization
3. Loop
4. Jump
5. Mutually exclusive sequences
6. Local synchronization



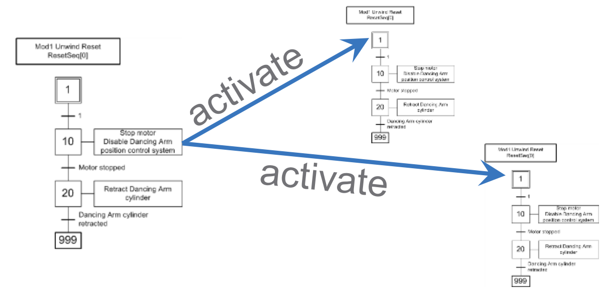

Fabrizio Avantaggiato
avafab@gmail.com

# Advanced design structures

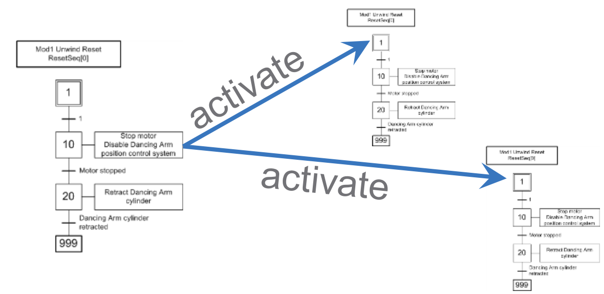
1. Macro steps
   - Steps which enclose a whole sequence, once activated they perform their own internal sequence




2. Father-child model
   - A “father” step in the main sequence activates “child” sub sequences that evolve regardless of the main sequences



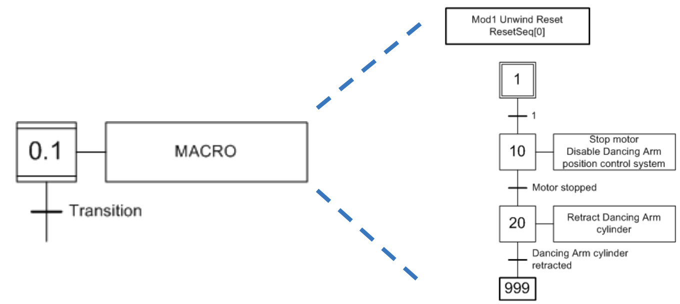

3. Supervision model
   - A supervisor step in the main sequence activates sub-sequences that evolve on their own until they finish or they are killed by the main sequence
   - The main supervision sequence can “kill” sub-sequences deactivating all their steps at any time



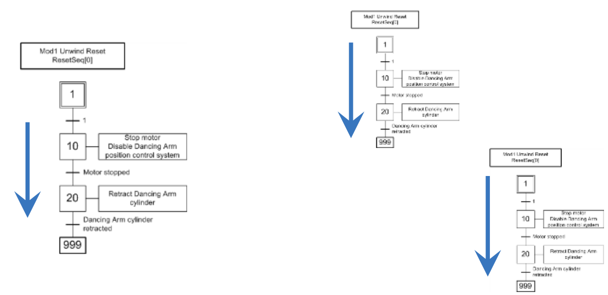

Fabrizio Avantaggiato
avafab@gmail.com

# Advanced design structures

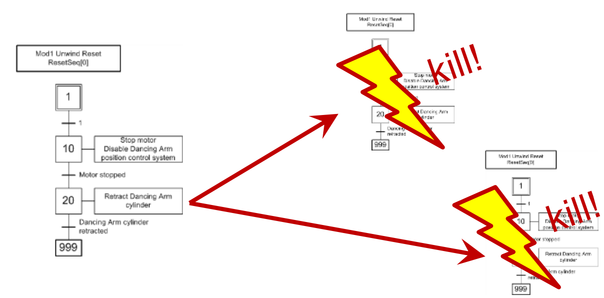
1. Macro steps
   – Steps which enclose a whole sequence, once activated they perform their own internal sequence




2. Father-child model
   – A “father” step in the main sequence activates “child” sub sequences that evolve regardless of the main sequences




3. Supervision model
   – A supervisor step in the main sequence activates sub-sequences that evolve on their own until they finish or they are killed by the main sequence
   – The main supervision sequence can “kill” sub-sequences deactivating all their steps at any time




Fabrizio Avantaggiato
avafab@gmail.com

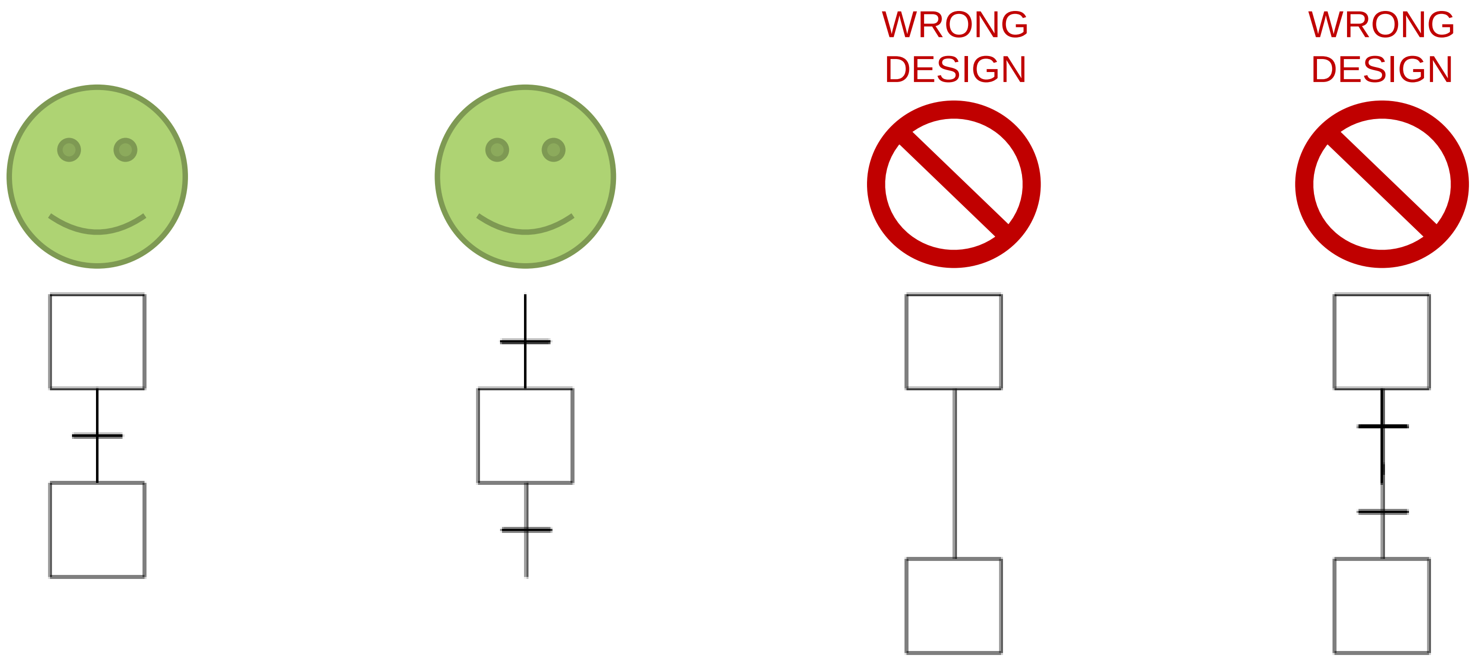
## Examples of wrong design

Classic mistakes on designing of an SFC:

1. Connecting steps (transitions) together without a transition (step) in between
2. Close a choice structure with a synchronization
3. Close a parallelism with a convergence
4. Modeling complex sequences using only one SFC

Fabrizio Avantaggiato
avafab@gmail.com

# Connecting steps (transitions) together without a transition (step) in between




Directed links connect steps and transitions with each other

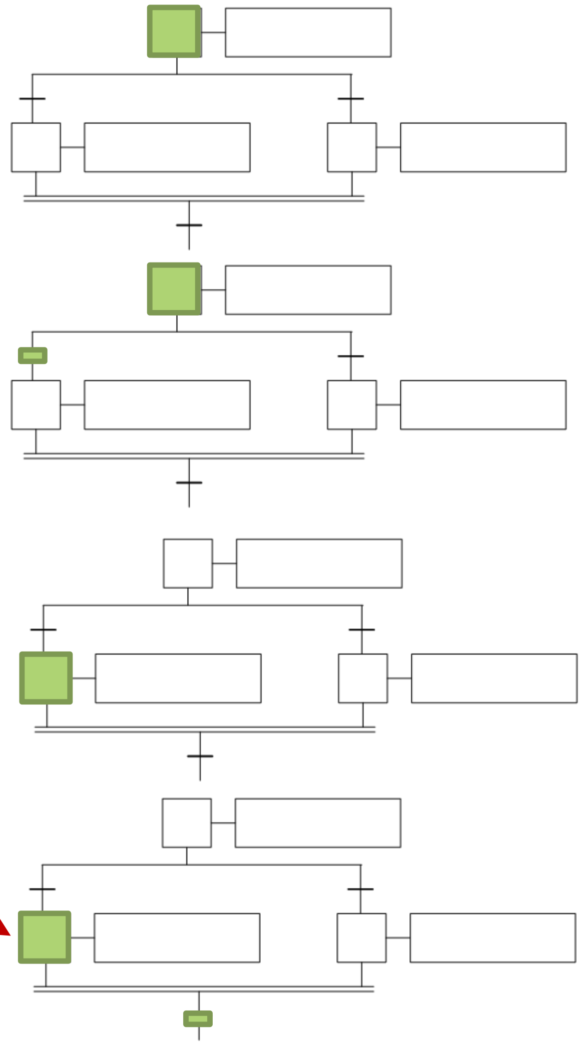
Direct links between two steps and between two transitions are not allowed.

Fabrizio Avantaggiato
avafab@gmail.com

# Wrong design patterns - Closing a choice structure with a synchronization

Make a choice

Close with a synchronization



Deadlock!

Machine will stay in this step forever

WRONG DESIGN

Evolution rule:

Steps are activated when both the following conditions are true:

- all steps above a transition are active ← Missing condition
- the connecting transition is superable (its conditions are all true)

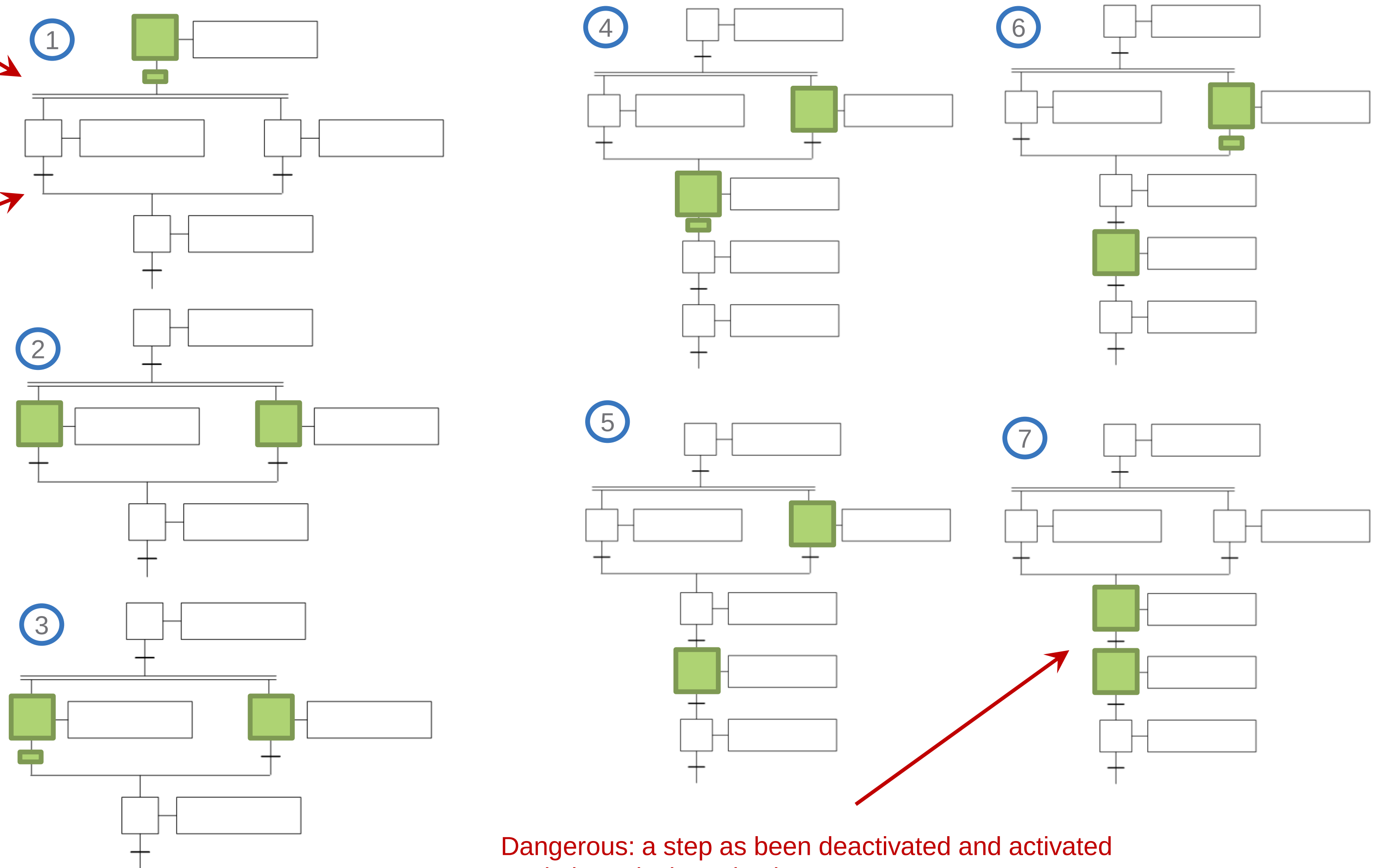
Fabrizio Avantaggiato
avafab@gmail.com

# Wrong design patterns - Closing a parallel structure with a convergence

Parallel structures

Close with a convergence

WRONG DESIGN




Dangerous: a step as been deactivated and activated again by a design mistake!

Unpredictable: in the same linear sequence we have actions in parallel

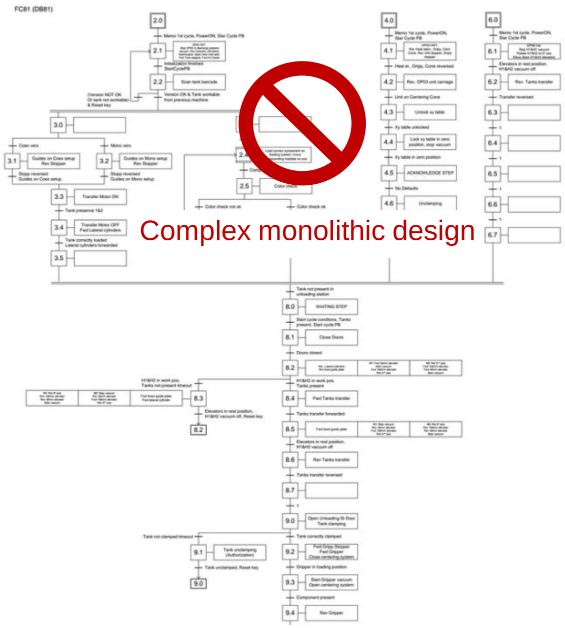
The model doesn’t behave as aspected

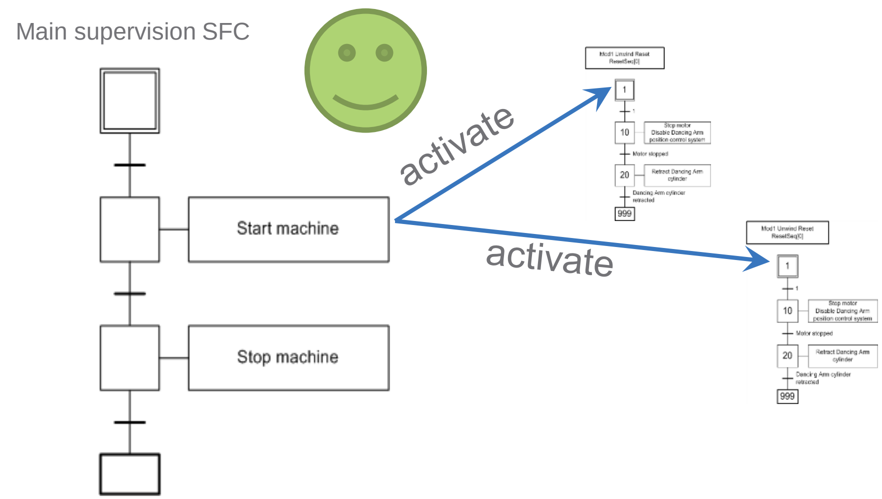

Fabrizio Avantaggiato

avafab@gmail.com

# Modeling complex sequences using only one SFC

- Use a supervision model when state machine complexity increases
  1. Split the whole process in independent subsequences
  2. Create a main supervisor SFC
  3. Link the subsequences with the main supervisor

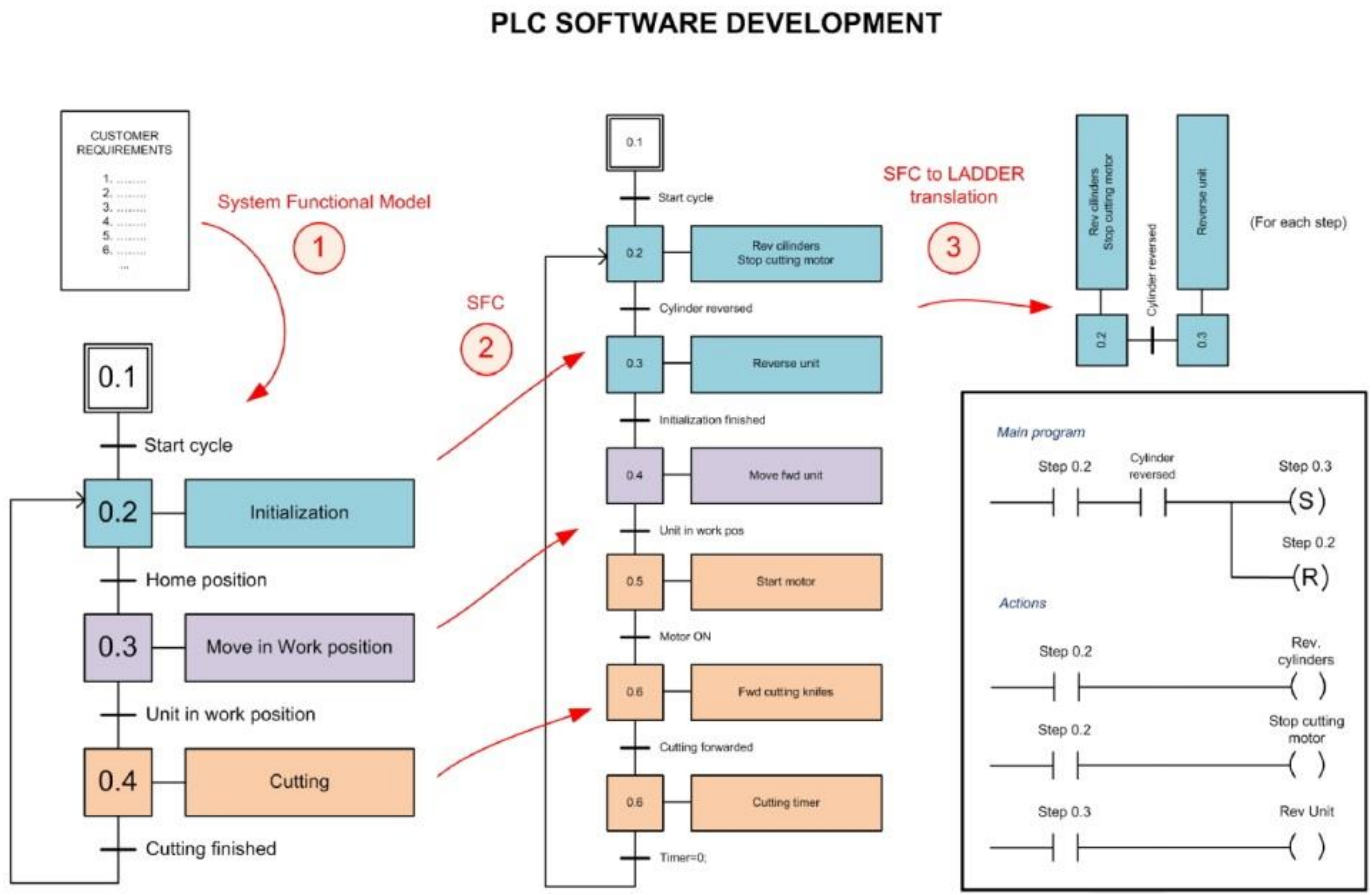




Fabrizio Avantaggiato
avafab@gmail.com

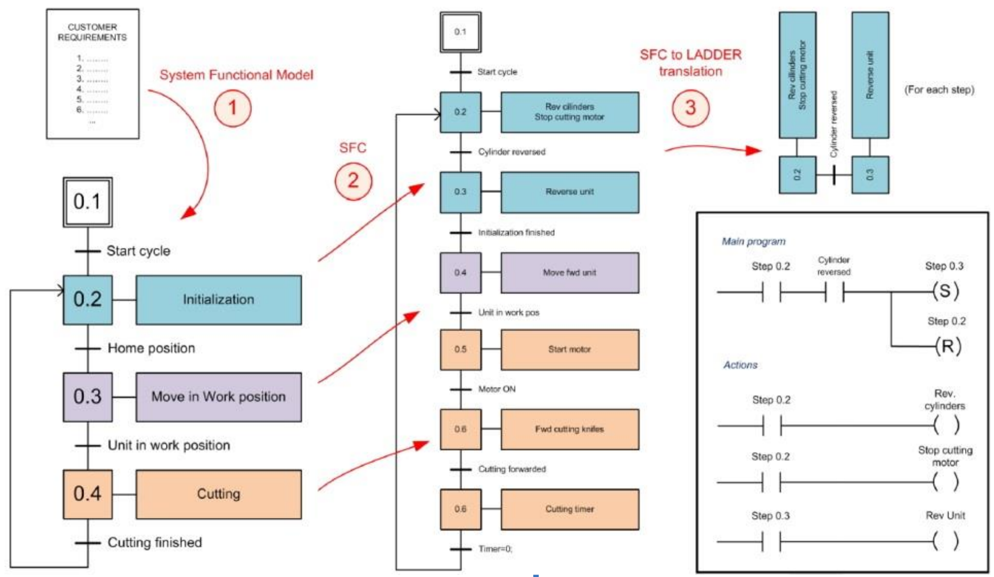
# From requirements to machine code using SFC model




Fabrizio Avantaggiato
avafab@gmail.com

# From requirements to machine code using SFC model

**PLC SOFTWARE DEVELOPMENT**




## ① System functional model

Capture client requirements and prepare and initial functional SFC with macro steps

Each macro step enclose a series of operations

## ② SFC

Split each macro step in its own operations

Specify all step actions and all conditions on transitions

## ③ SFC to LADDER translation

- 1 routine to describe SFC evolution rules
- 1 routine to call the actions linked to each step

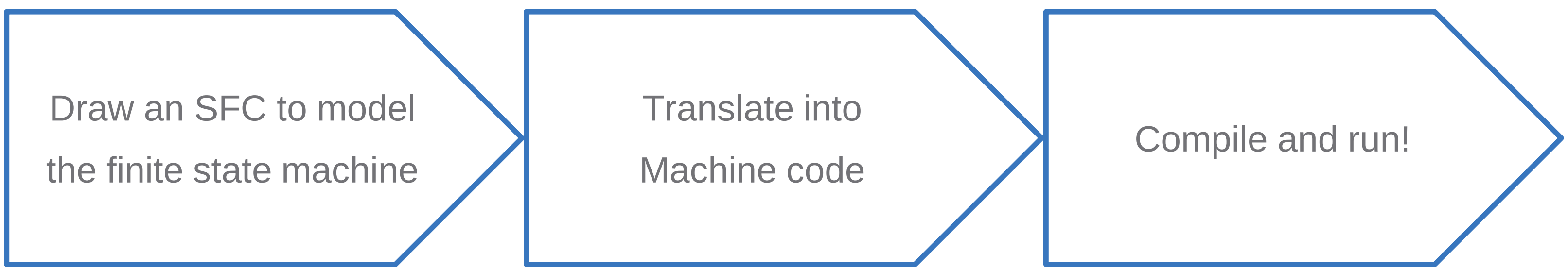
Fabrizio Avantaggiato
avafab@gmail.com

## Why modeling a finite state machine with SFC

SFC has been defined as a standard language to program industrial logic controllers and modern PLC manufacturers have recently introduced it as a programming language in their tools.

Software developers can now drag and drop structures, compile and download the diagrams into the PLC, without writing a single line in machine code.

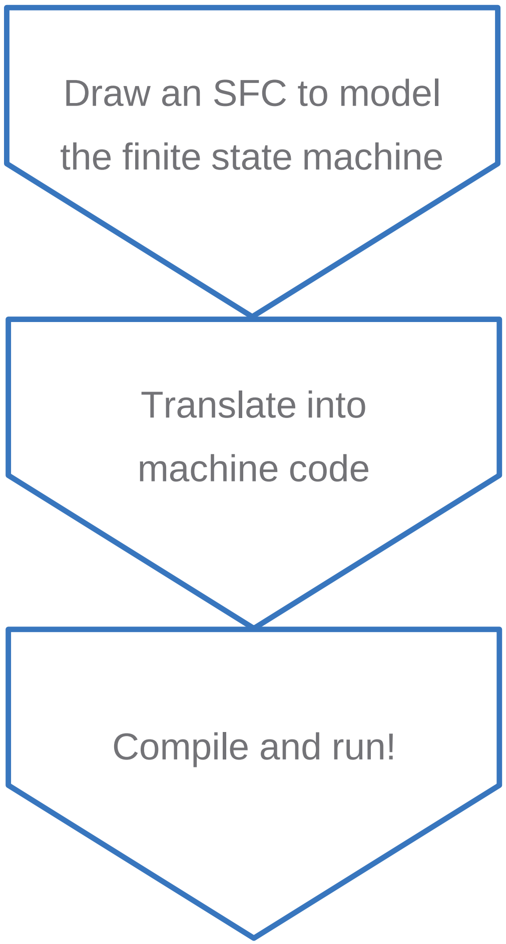
➢ But SFC is basically a modeling language derived from a mathematical concept, this means it is suitable to design any kind of process that can be split in a sequence of steps (finite state machine) independently from which kind of hardware and software is being used.



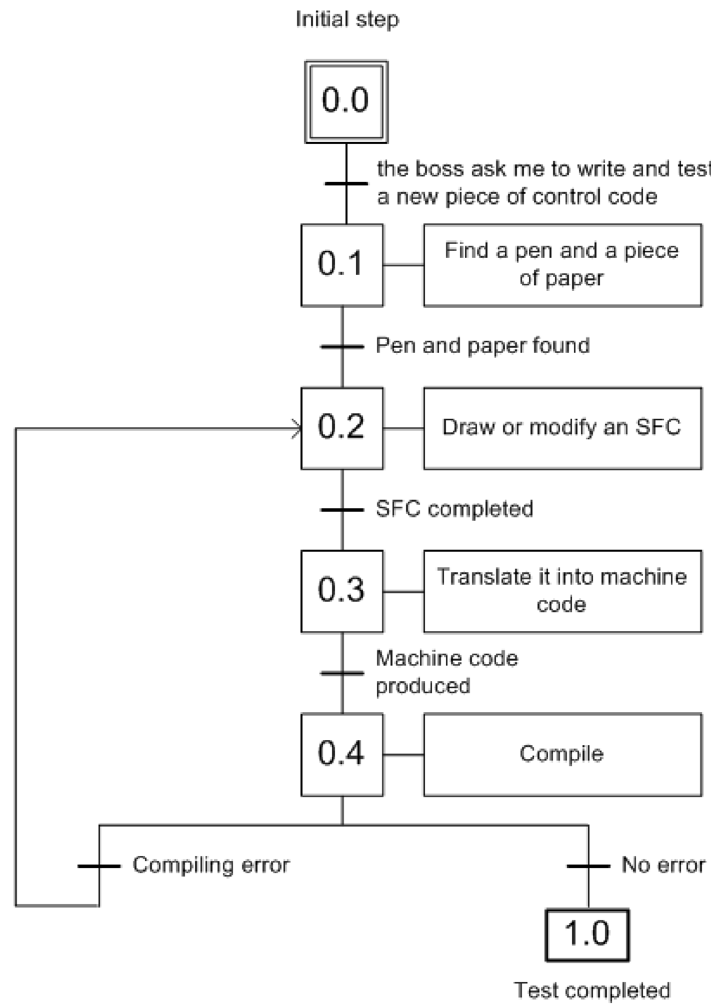

Fabrizio Avantaggiato
avafab@gmail.com

# A simple example?

My process



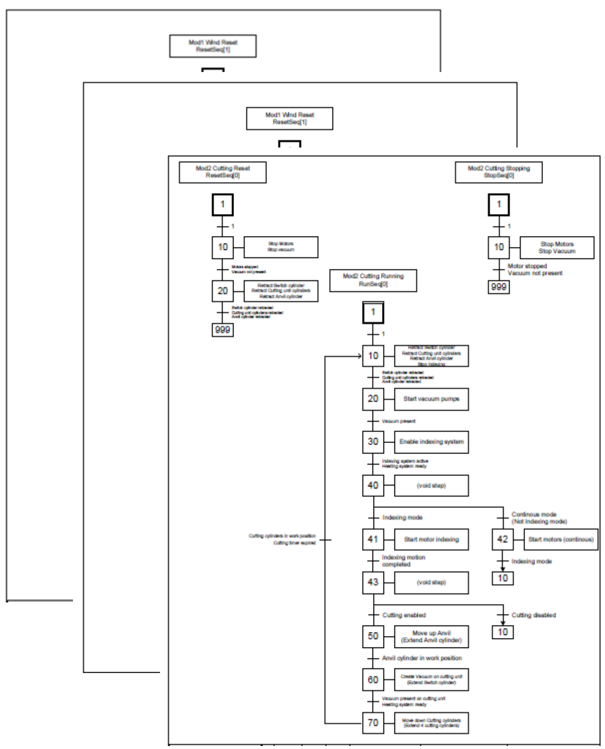

My process’ SFC model

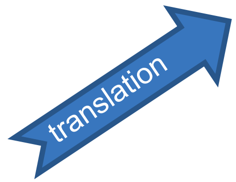


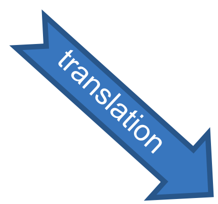

Fabrizio Avantaggiato
avafab@gmail.com

# A real industrial example

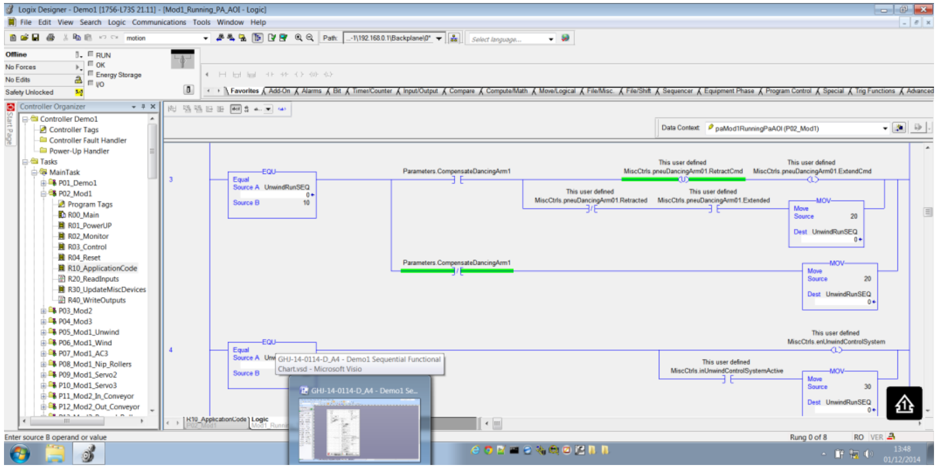
Microsoft Visio with custom SFC library



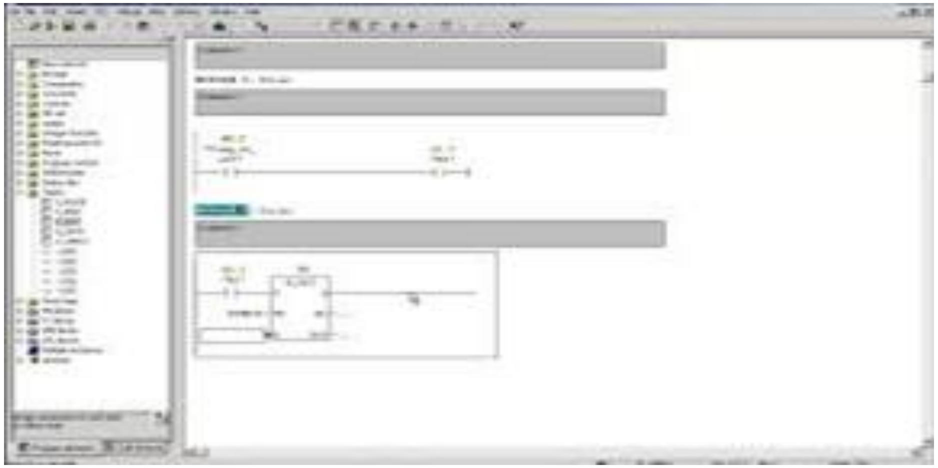




Rockwell PLC - Logic Studio5000



Siemens PLC -  Step7



Fabrizio Avantaggiato
avafab@gmail.com

# SFC modeling benefits

- **Understanding and put requirements on the paper**
  - SFC is easy to read.
    - ✓ SFC model can be used as a shared document to assure clear information exchange between software, electrical and mechanical teams.
    - ✓ Due to its similarity to a flow chart, SFC can be easily read by non technical people too.

- **Reducing commissioning time**
  - Automatic sequences can be easily validated on the paper before any line of code is written.
  - Machine code derived from it can be quickly verified on the machine to be sure the mechanics behaves according with the model.

- **Helping maintenance operations**
  - In case of a machine blockage, maintenance team can use the SFC to find out which conditions are preventing the machine to evolve into the next state.

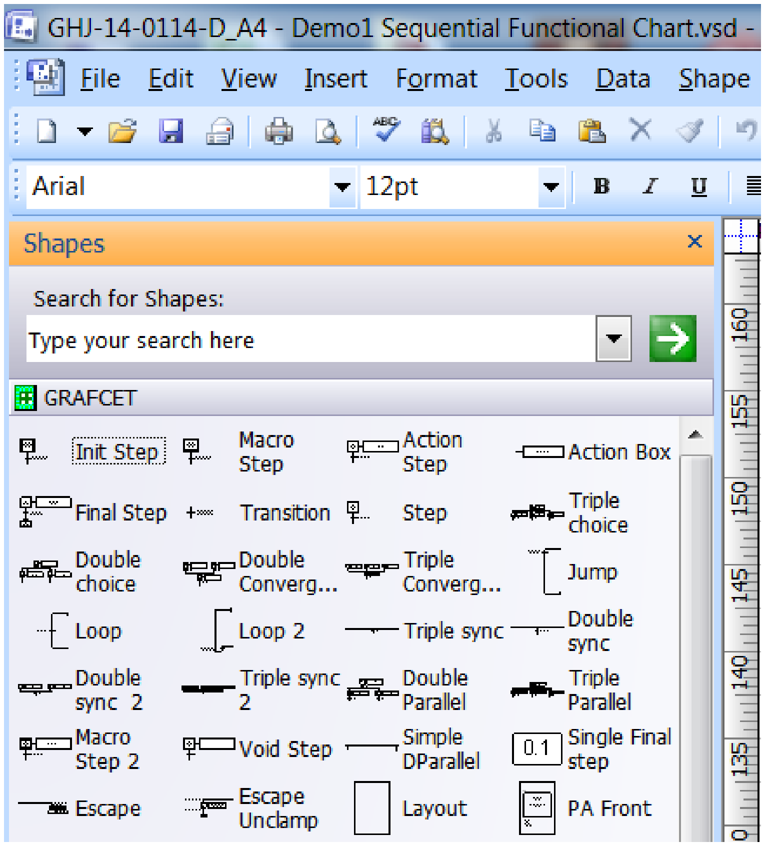
- **Reducing time to read and modify existing machine code**
  - Any modification can be easily made and tested on the paper and then quickly translated into machine code

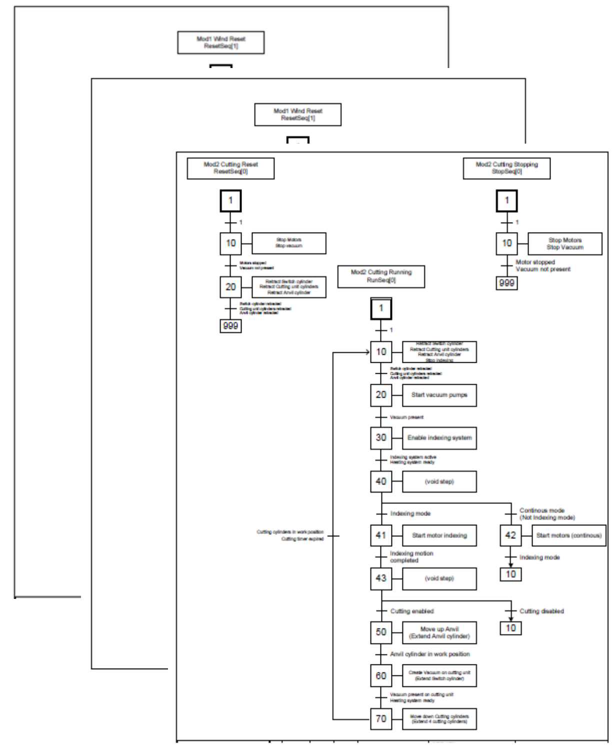
- **Platform independent**
  - SFC can be used to model any finite state machine and then translated in any language

Fabrizio Avantaggiato
avafab@gmail.com

# Modeling tools

## Microsoft Visio SFC Library – Developed by Fabrizio Avantaggiato



Fabrizio Avantaggiato
avafab@gmail.com

# Bibliography


- Programming Industrial Control Systems Using IEC 1131-3
- Rockwell Logix5000 controllers using Sequential Function Charts
- SFC and translation to ladder examples

- Tecnologie informatiche per l'automazione industriale – Chiacchio/Basile (Italian only)