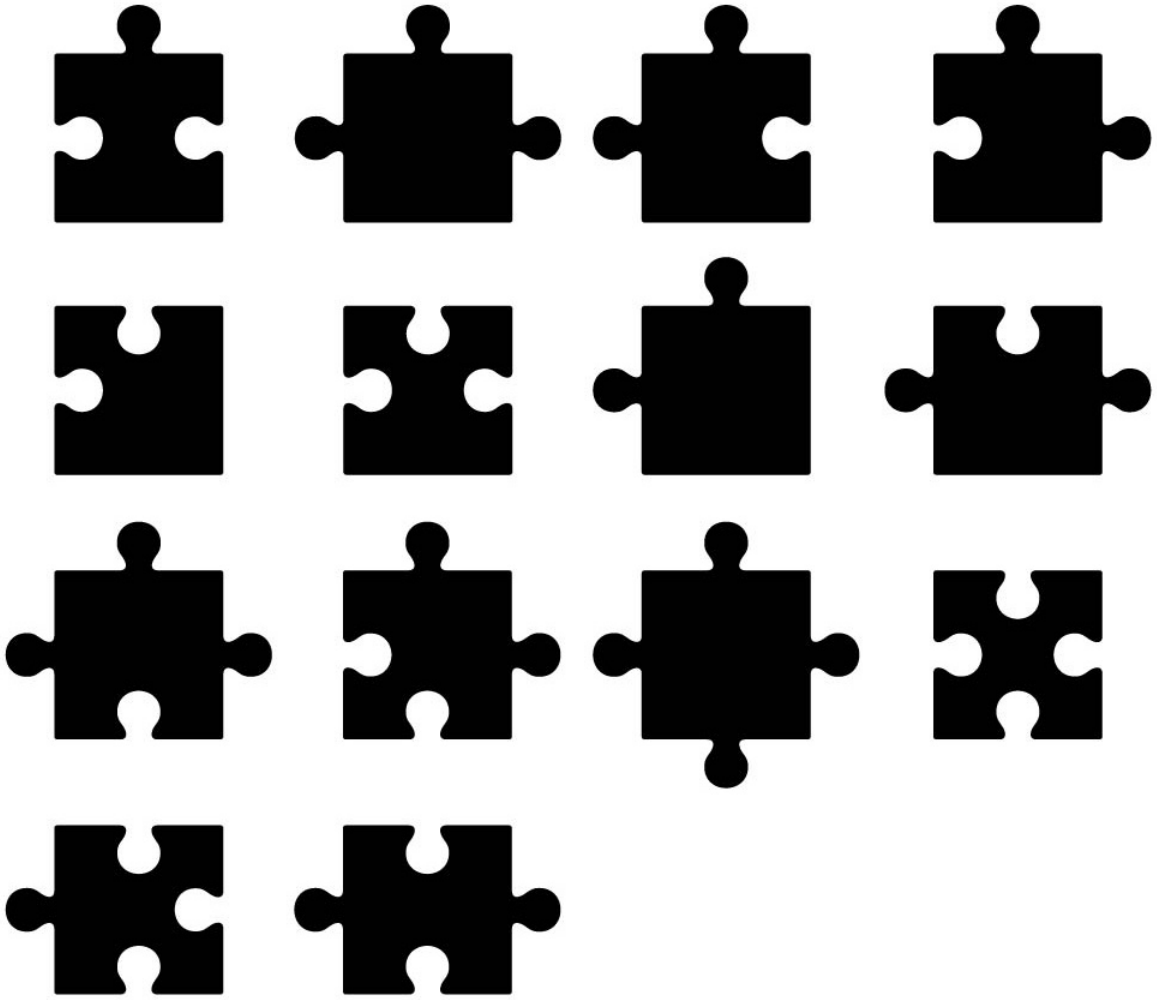
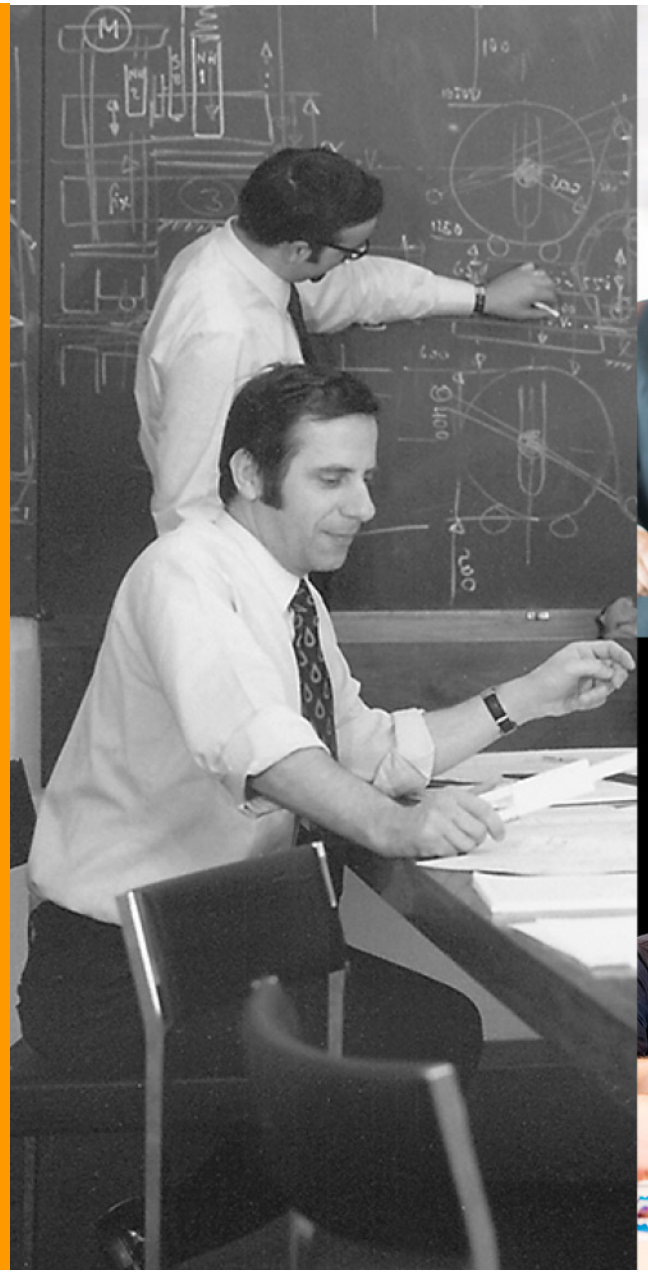


zühlke
empowering ideas

# Template Based Smart Contract Generation

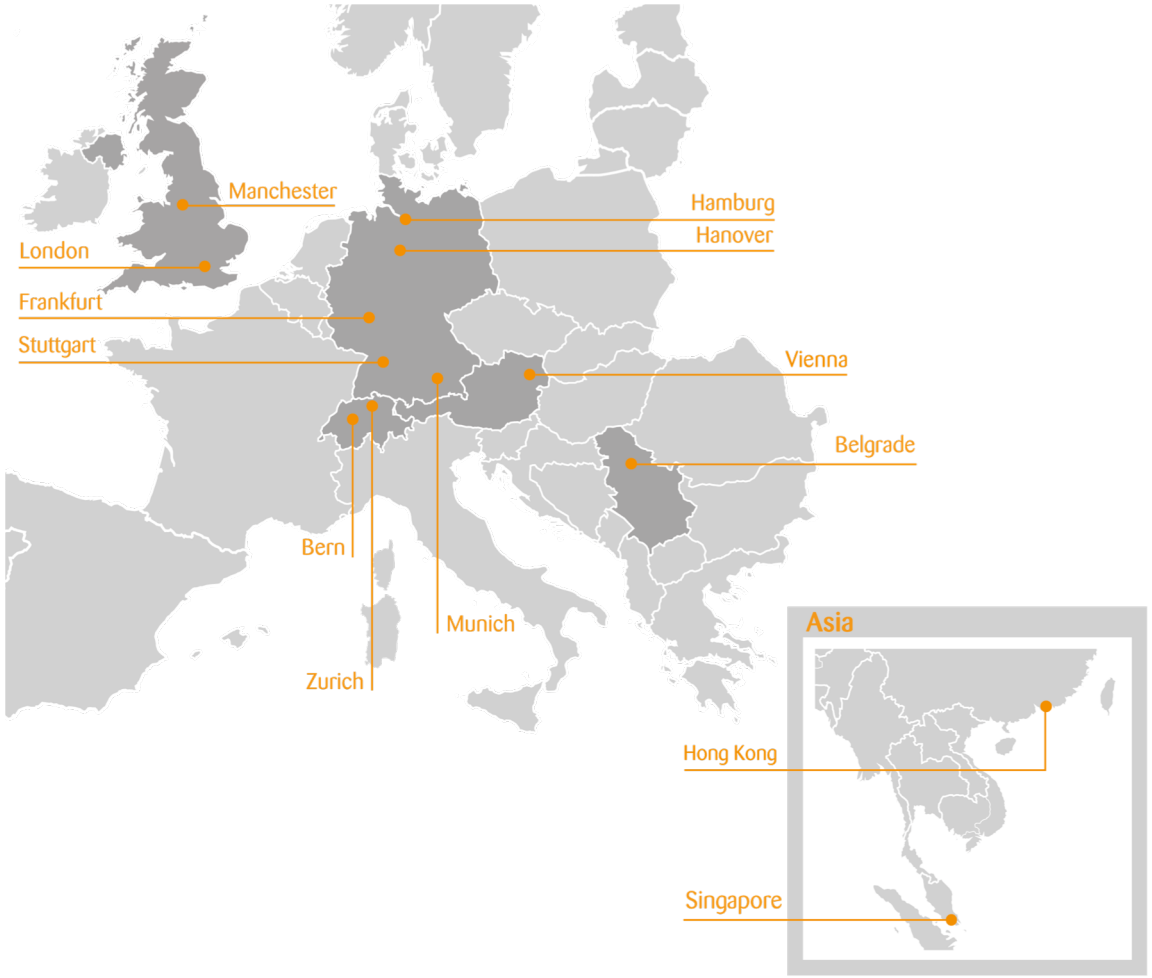
Dr. Klaus Alfert

As a partner for business innovation, we are committed to ensuring your success – yesterday, today, and in the coming years.




# About Zühlke

## Facts and figures




- Founded 1968
- Owned by partners
- Teams in Germany, United Kingdom, Austria, Serbia, Singapore, Hong Kong and Switzerland
- Over 10,000 projects implemented
- 1,000 employees and a turnover of CHF 154 million (2017)
- Certifications: ISO 9001 and 13485

# Smart Contracts are hard

https://medium.com/@peckshield/alert-new-batchoverflow-bug-in-multiple-erc20-smart-contracts-cve-2018-10299-511067db6536

```
function batchTransfer(address[] _receivers, uint256 _value) public returns (bool) {
    uint cnt = _receivers.length;
    uint256 amount = uint256(cnt) * _value;
    require(cnt > 0 && cnt <= 20);
    require(_value > 0 && balances[msg.sender] >= amount);

    balances[msg.sender] = balances[msg.sender].sub(amount);
    for (uint i = 0; i < cnt; i++) {
        balances[_receivers[i]] = balances[_receivers[i]].add(_value);
        Transfer(msg.sender, _receivers[i], _value);
    }
    return true;
}
```

# Can you explain or spot the bug?

https://medium.com/@peckshield/alert-new-batchoverflow-bug-in-multiple-erc20-smart-contracts-cve-2018-10299-511067db6536

```
function batchTransfer(address[] _receivers, uint256 _value) public returns (bool) {
    uint cnt = _receivers.length;
    uint256 amount = uint256(cnt) * _value;
    require(cnt > 0 && cnt <= 20);
    require(_value > 0 && balances[msg.sender] >= amount);

    balances[msg.sender] = balances[msg.sender].sub(amount);
    for (uint i = 0; i < cnt; i++) {
        balances[_receivers[i]] = balances[_receivers[i]].add(_value);
        Transfer(msg.sender, _receivers[i], _value);
    }
    return true;
}
```

# Can you explain ~~or spot~~ the bug?

# Coding a Smart Contract is hard

- New programming languages (e.g. Solidity, BitCoin-Script)
- Unusual execution platforms
- New bug patterns
- New and unreliable toolings

# How to enable business people to leverage the computer by themselves?

# The early years

# The first revolution 1959: Grace Hopper invents COBOL!

```
Identification Division.
Program-ID. HELLOPGM.
Procedure Division.
Display "Hello World!".
STOP RUN.
```

Smithsonian Institution, Wikipedia, CC BY 2.0

Wikipedia, CC 3.0

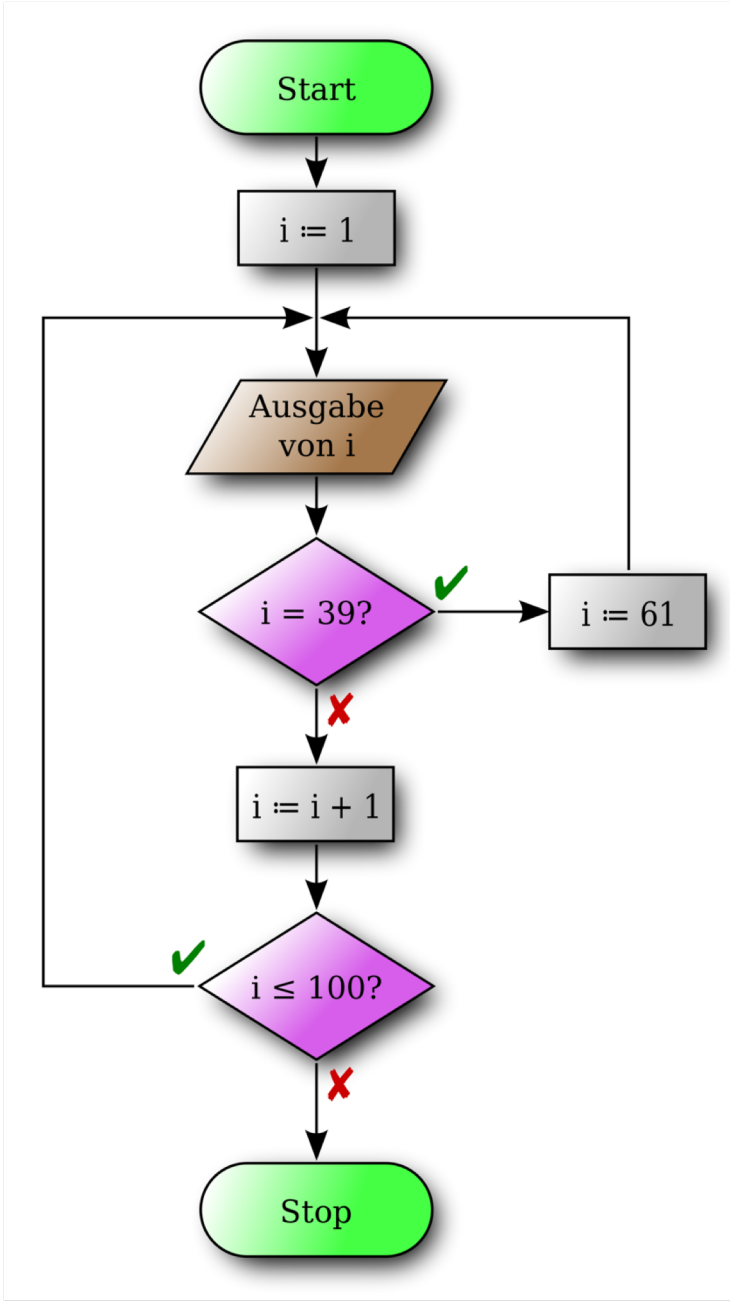
# The 2nd revolution 1975: SQL developed at IBM

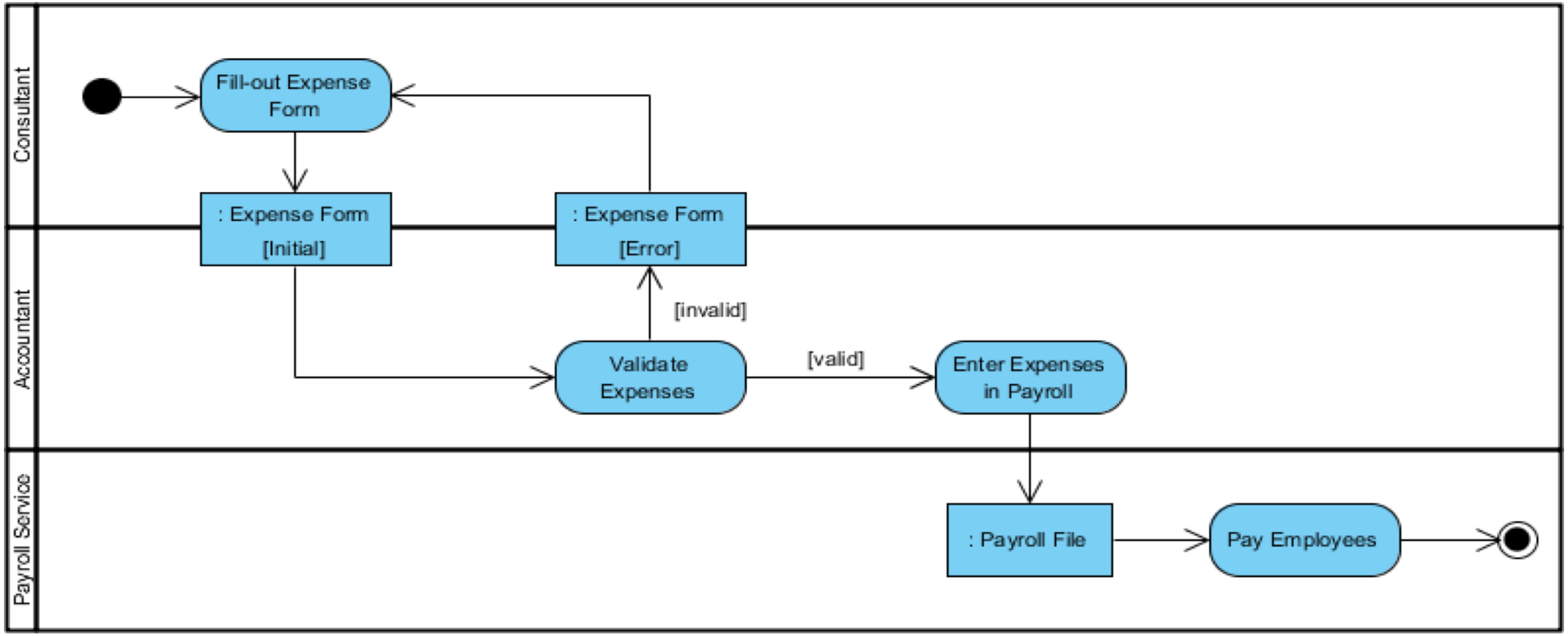
```
SELECT *
FROM SALES
WHERE CUSTOMER = 123
```

# Graphical Models

https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Flowchart_de.svg/285px-Flowchart_de.svg.png

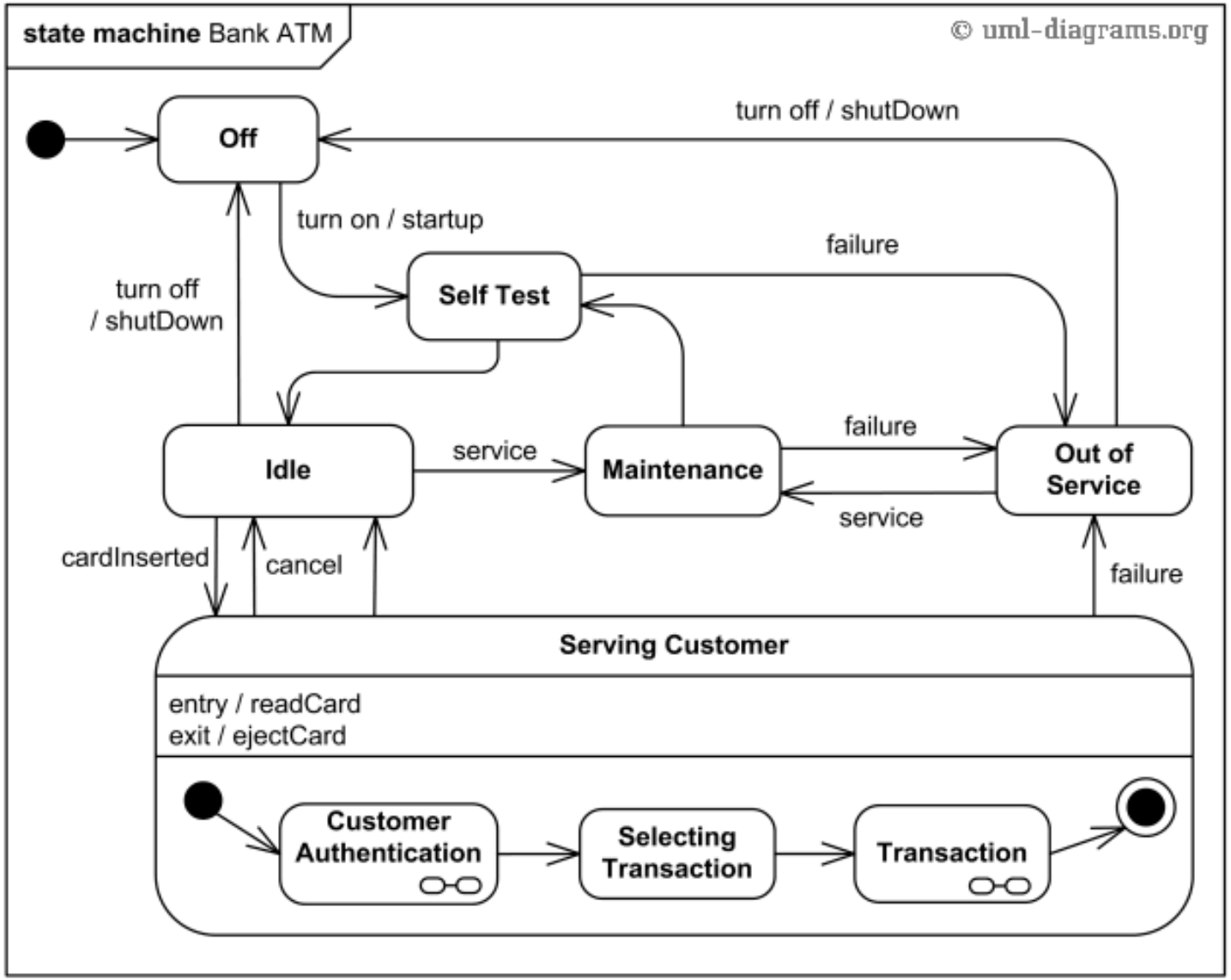
# Flow Charts: Modeling Gotos since the 50ies – and natively in Powerpoint ☺

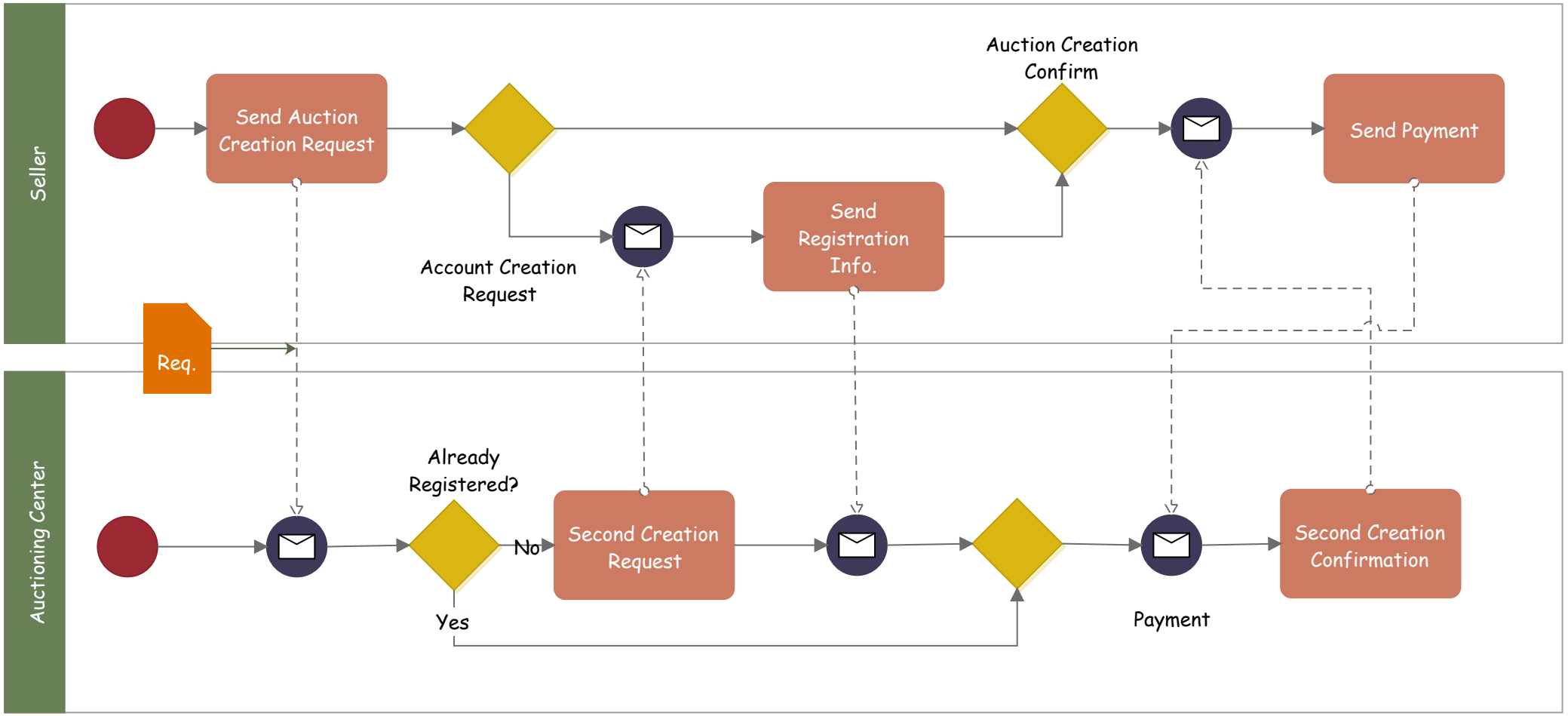





# The ubiquitous UML: Activity Diagrams

# The ubiquitous UML: Statechart Diagrams

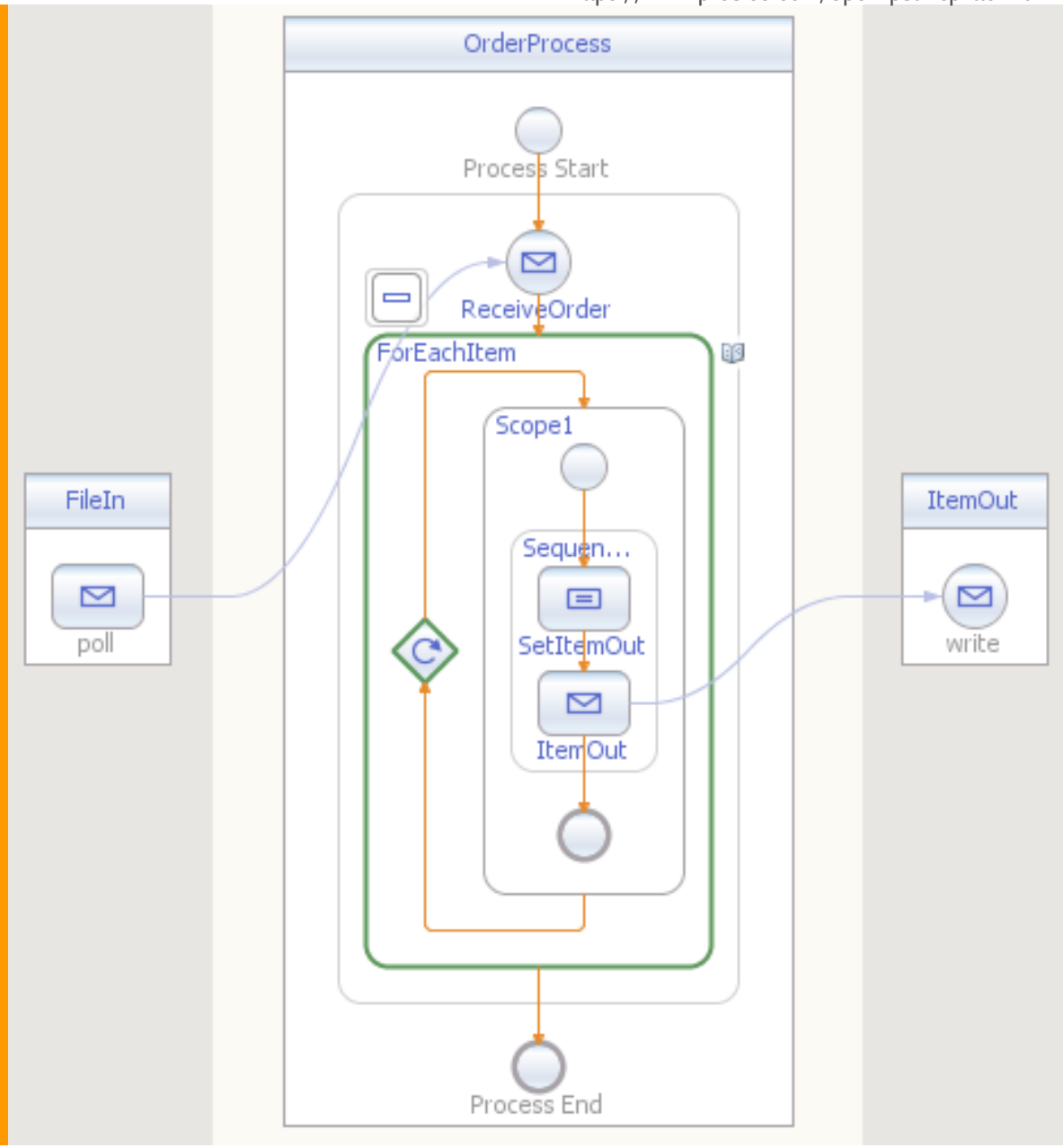




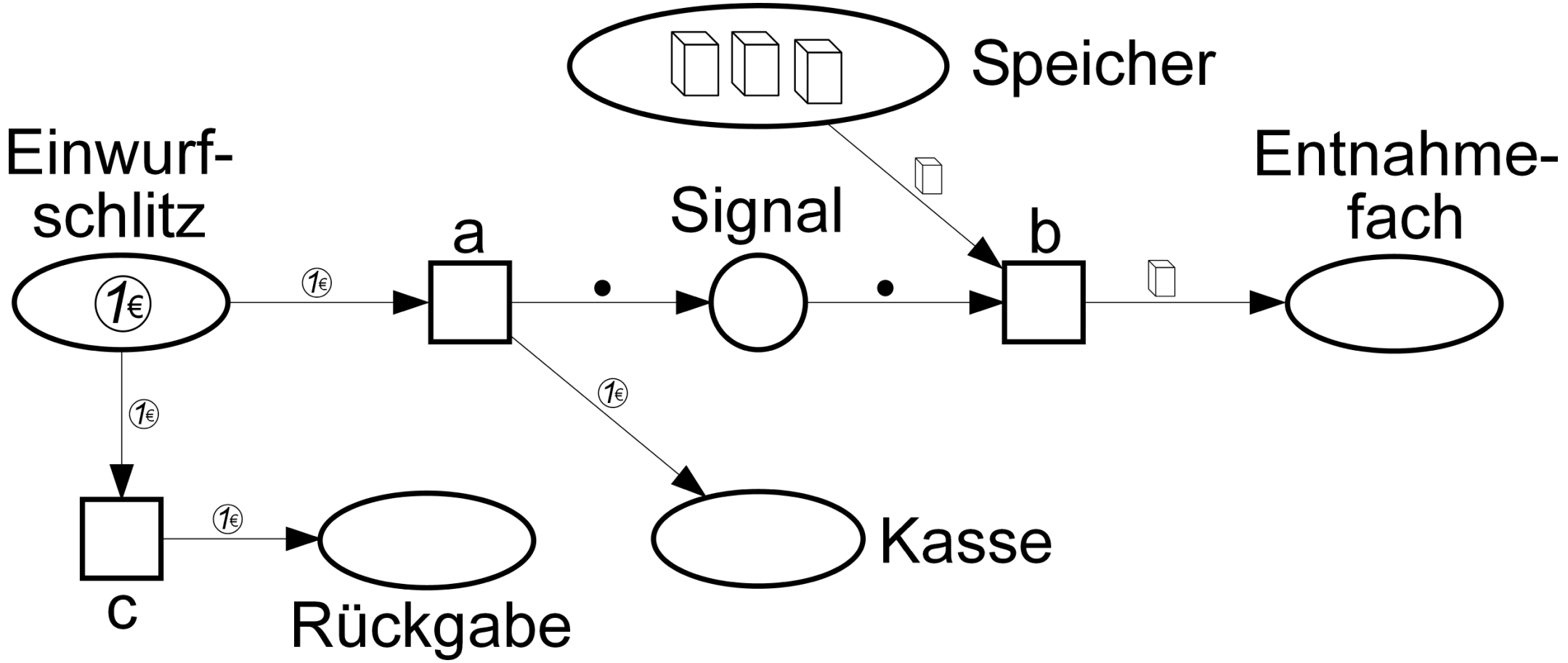

# BPMN – Business Processes Modeling Notation

https://www.predic8.com/bpel-xpath-splitter.htm

# BPEL: Scripting of Web Services in SOAs



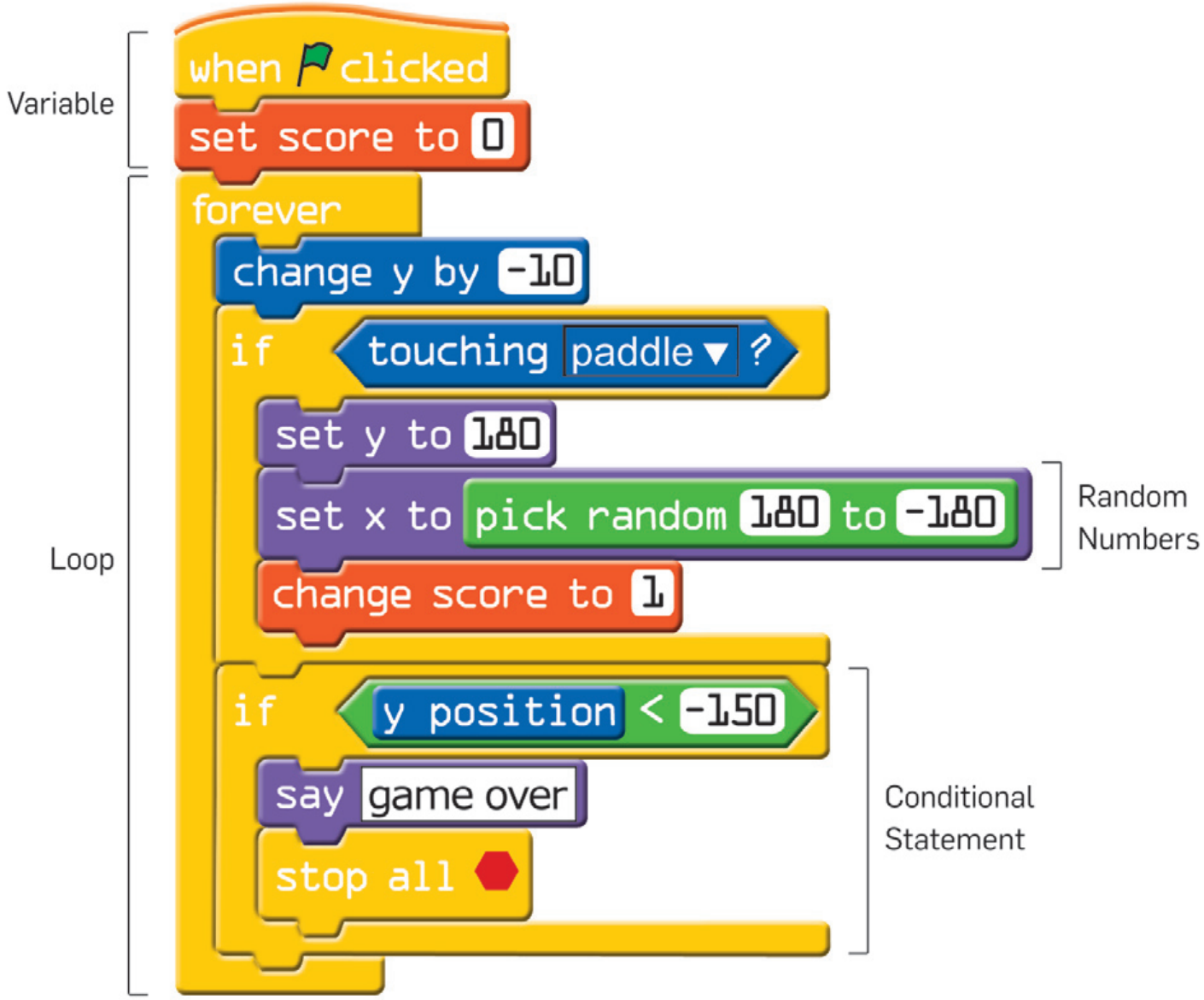

 



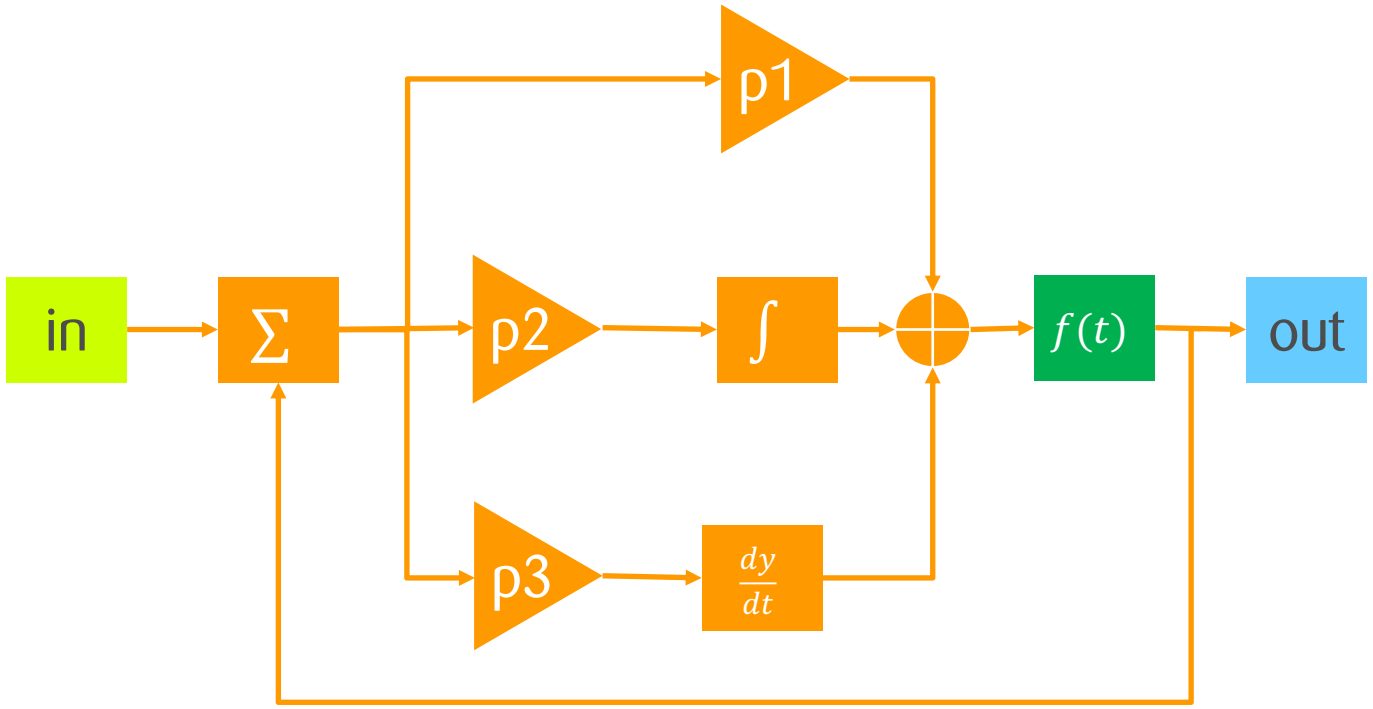

# Petri Net:
# Modeling concurrent & discrete systems since 1962

# Scratch: Programming for All (8 years and older)






# Block diagram of PID controller in a feedback loop

# Observations


## Action Sequences & Control Flow

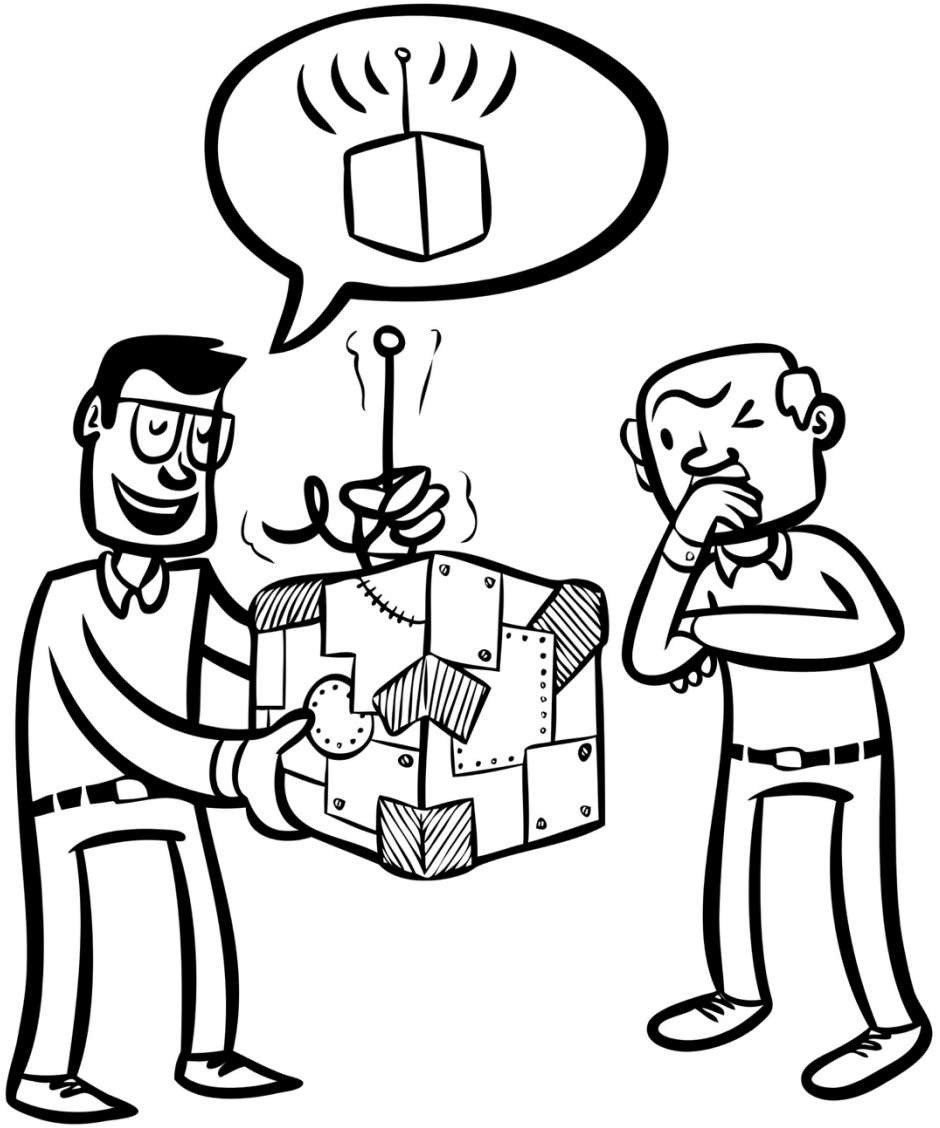
- Flow Charts are compelling, but don't scale and have no structure
- UML, BPMN abstract away a lot of the details: Good for discussion, bad for execution
- BPEL: designed around web services, does not really fit.
- Petri Nets: Formally sound, but (sometimes) difficult to comprehend, few structuring elements
- Scratch is easy, even for kids and thus for business and legal people, but it is a real programming language

## Data Flow

- Block diagrams: ubiquitous in engineering, unusal in a business context

# How to enable business people to leverage the ~~computer~~ BLOCKCHAIN by themselves?

# Prototyping and experiments show the path to enlightment.

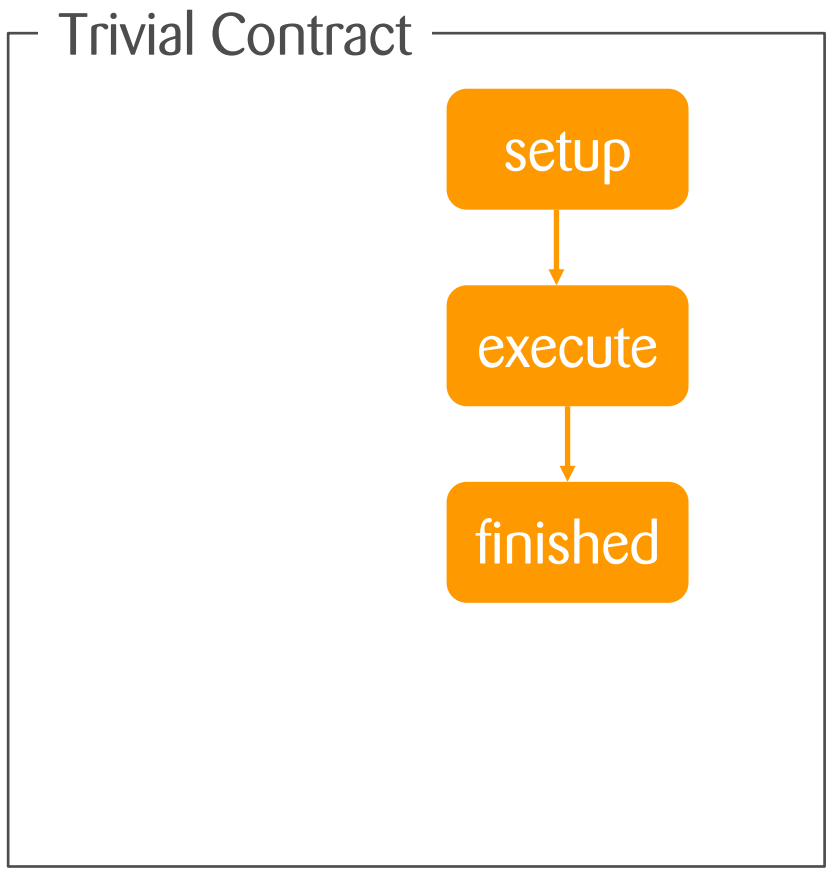


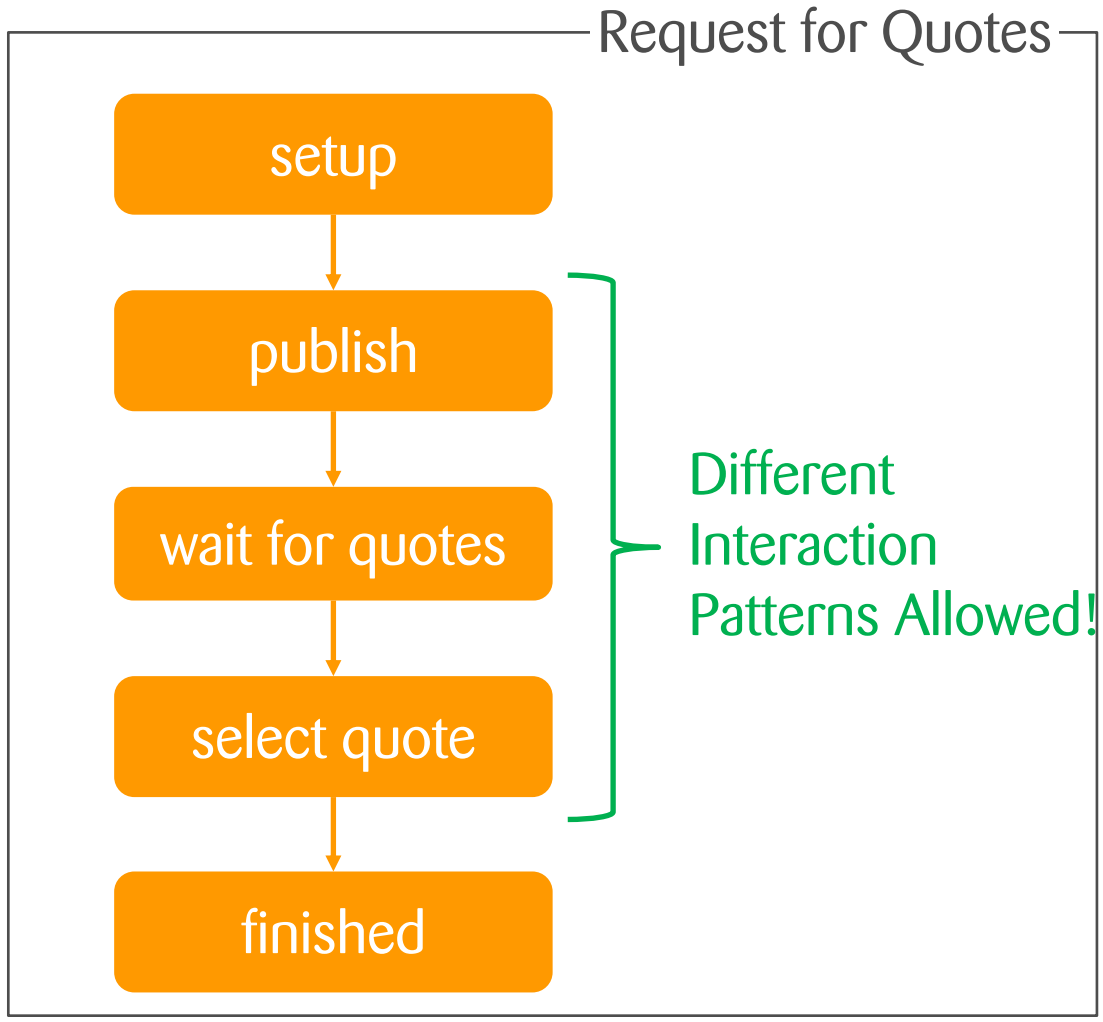
# Guy Steele, Jr: „Growing a language“

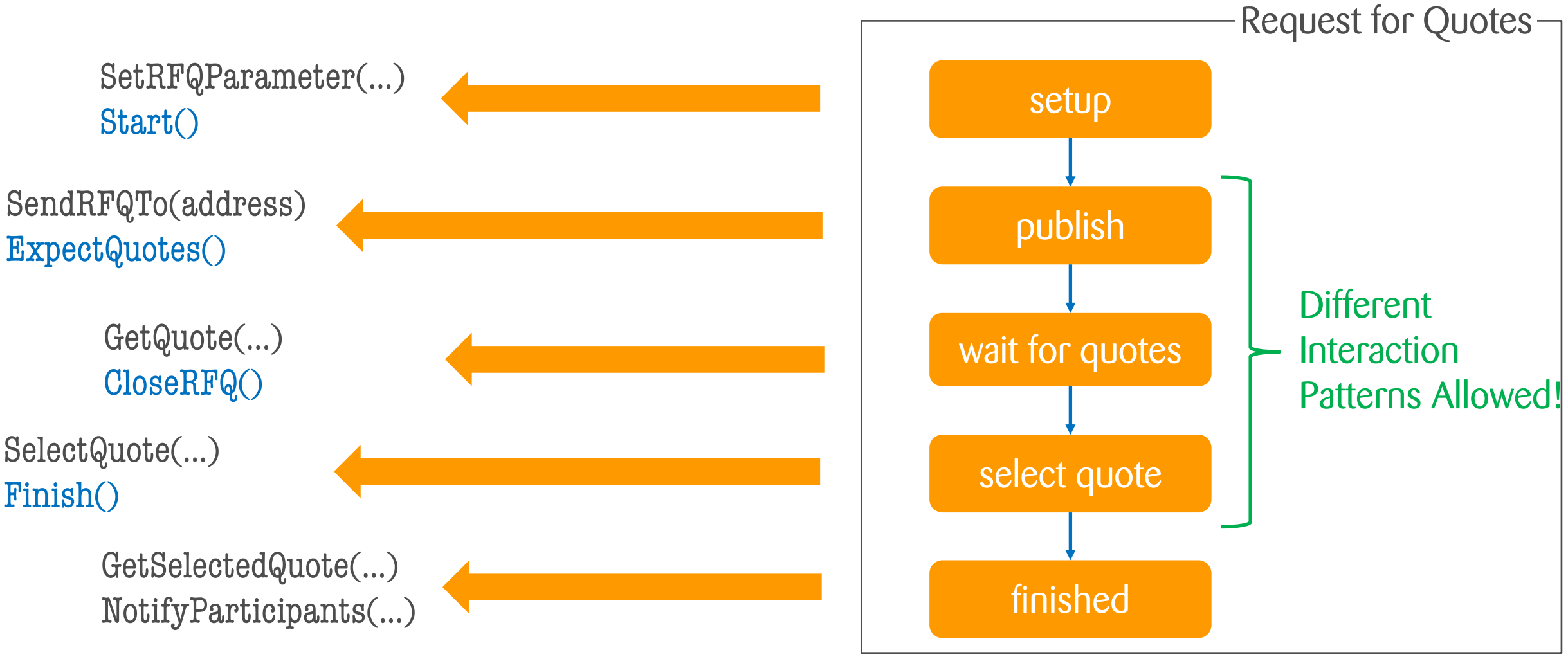
# Few elements, carefully selected and composable

# Contracts operate in different states



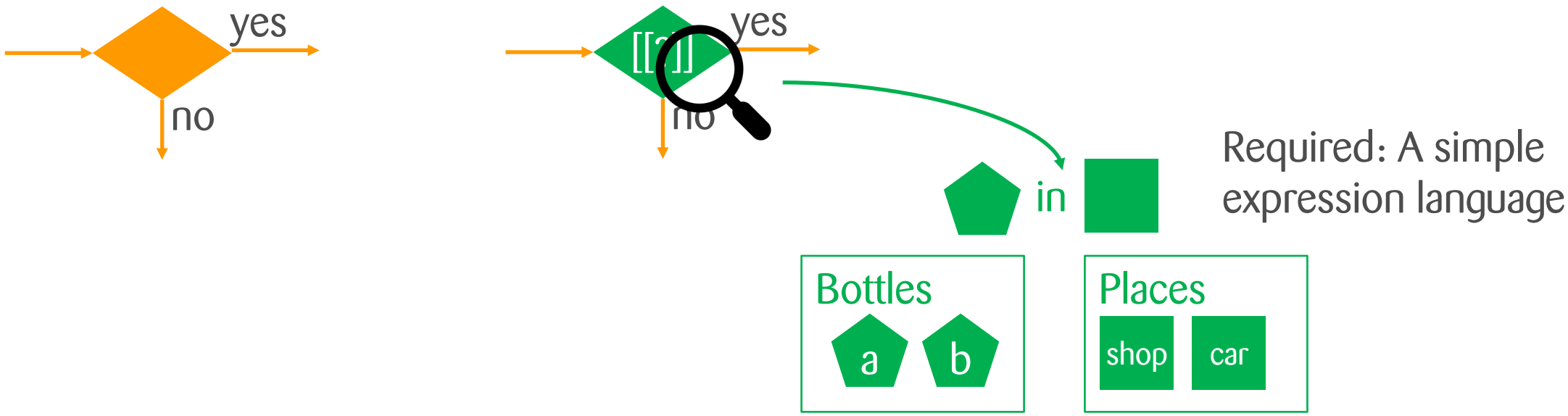



# Contracts operate in different states



# Also well-known symbols need details

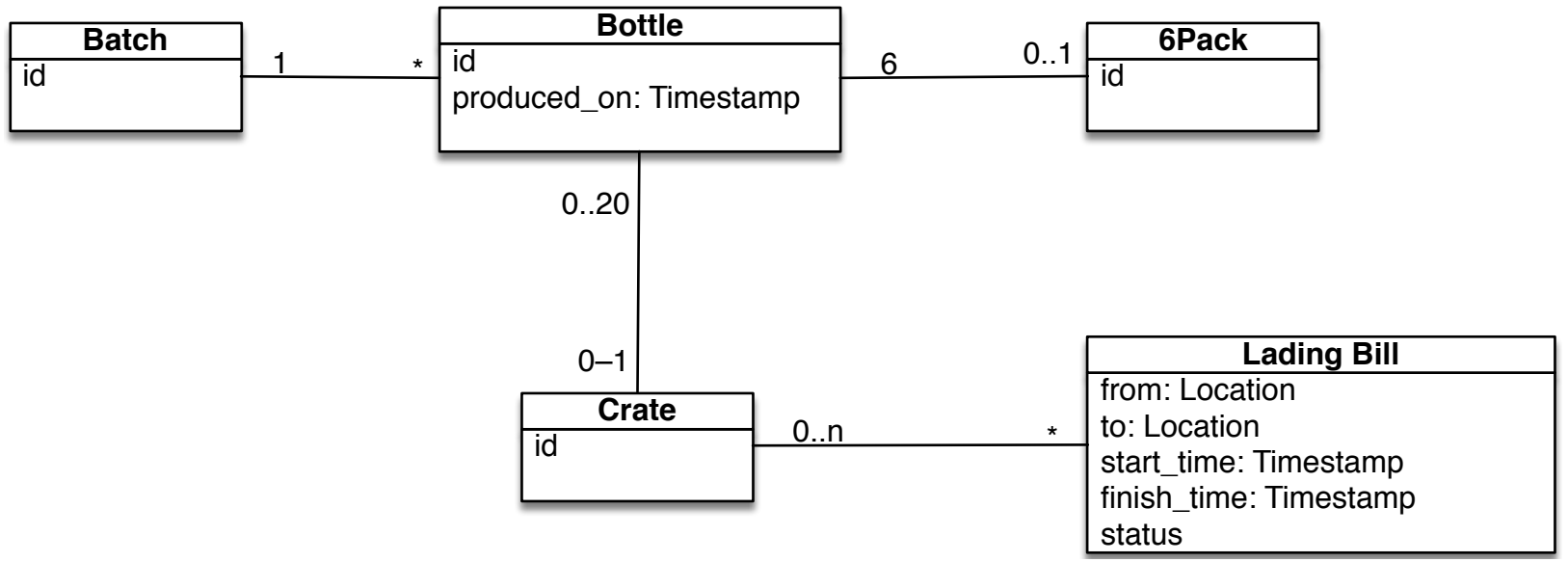
- On which data do we operate?
- What is the semantic of an operator?




Required: A simple expression language

# On what data operates the contract?

- For storing items in a ledger
  - dependencies between items are relevant
  - Attributes and their values of items are relevant
- An assignment operation to create and assign values or expressions
  - Reuse the expression language
  - `let LadingBill.status = done`



# What is a Template? A Library? A Framework?

## Library

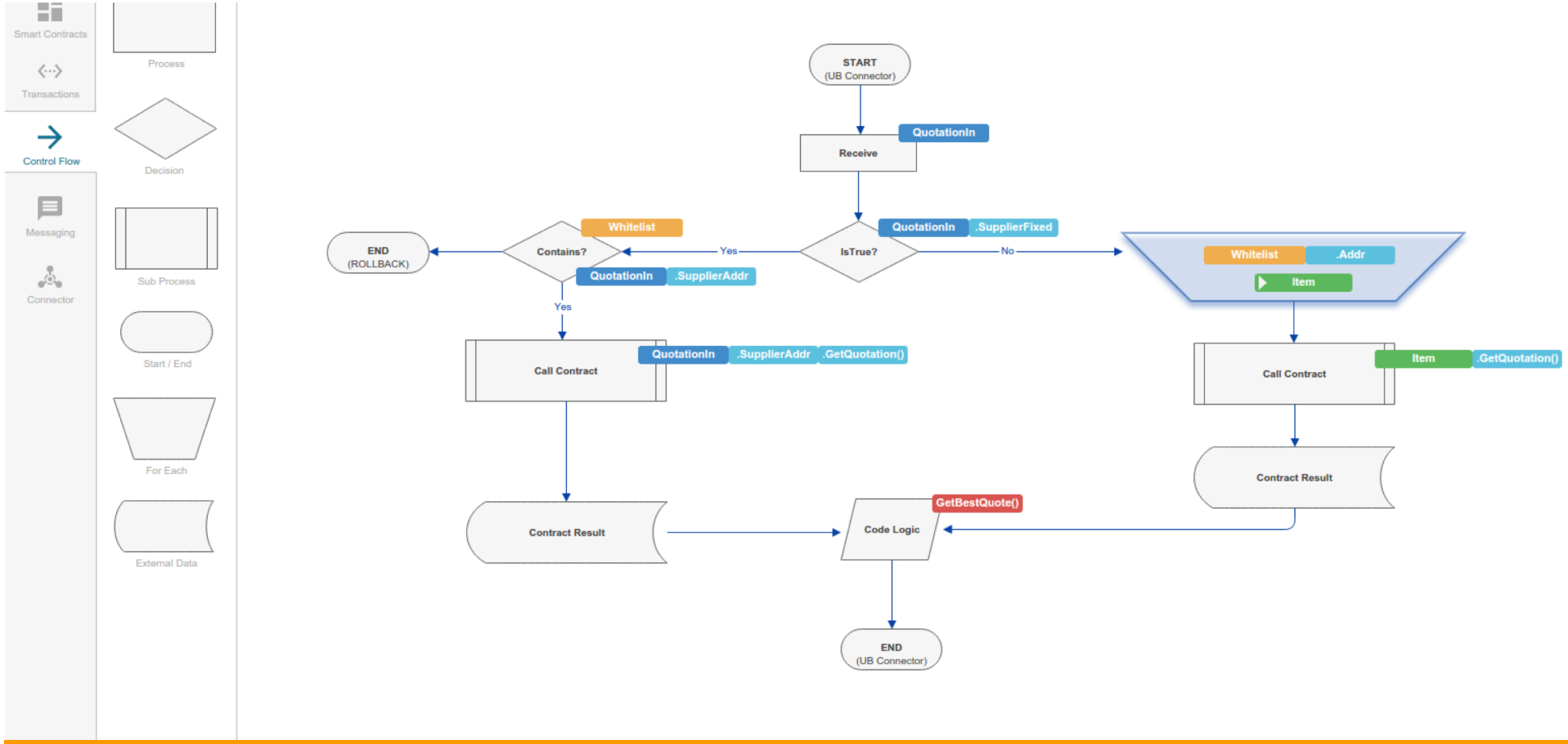
- Add-On to a system.
- The developer decides when the library functionality is executed.

## Framework

- The control flow lives inside the framework
- Hollywood-principle: don't call us, we call you!

## Template

- Framework, preferring configuration over coding

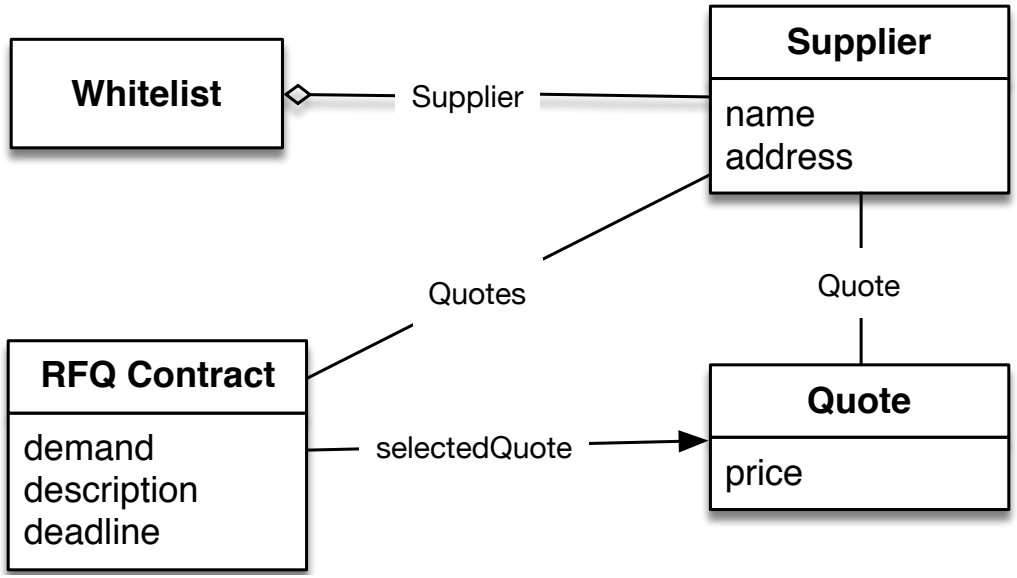



| | |
|---|---|
| Container | Whitelist |
| Attribute | [Address] |
| Enumerating Item | "Item" |

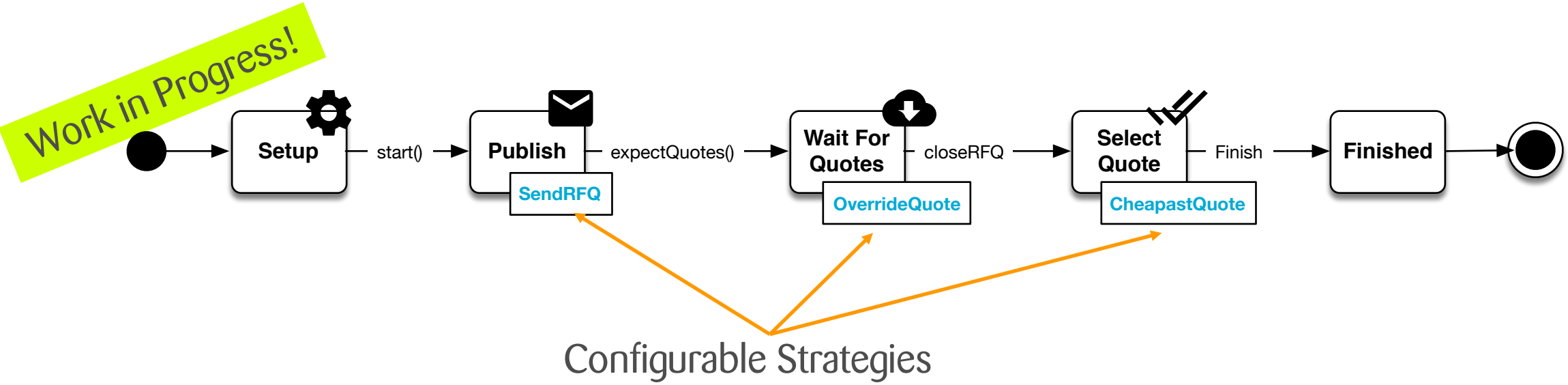
# The „original“ RFQ-process

## Unibright Technical Details (v04, March 2018)

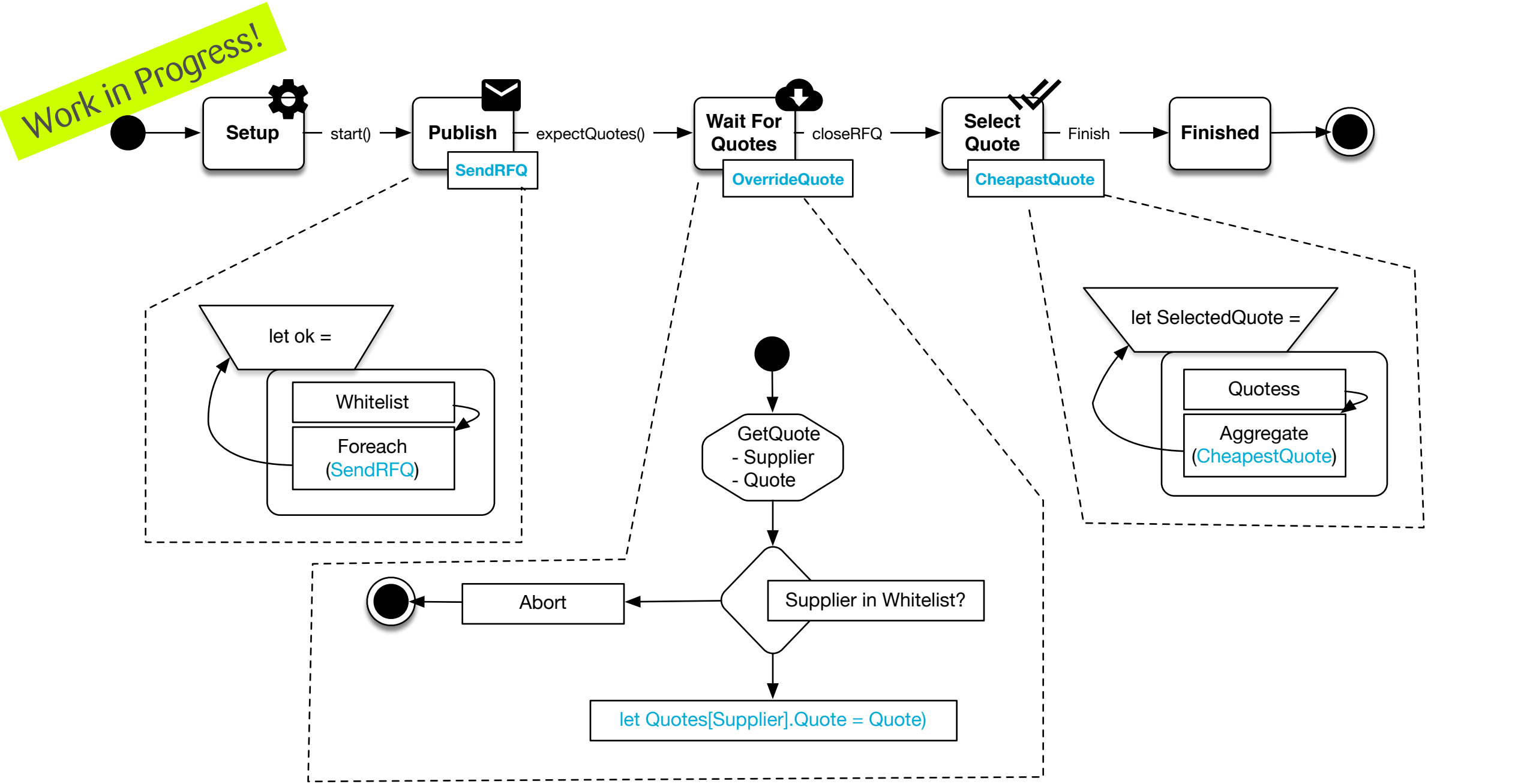
Work in Progress!


# RFQ Template May 2018
# States and Data




# RFQ Template May 2018
# Flow, Configuration, Details


Work in Progress!
Setup
start()
Publish
SendRFQ
expectQuotes()
Wait For Quotes
OverrideQuote
closeRFQ
Select Quote
CheapastQuote
Finish
Finished
let ok =
Whitelist
Foreach (SendRFQ)
GetQuote
- Supplier
- Quote
Abort
Supplier in Whitelist?
let Quotes[Supplier].Quote = Quote)
let SelectedQuote =
Quotess
Aggregate (CheapestQuote)

# Enjoy Blockchain Programming with Unibright Templates!

zühlke
empowering ideas

**Dr. Klaus Alfert**

+49 171 475 6817
klaus.alfert@zuehlke.com
@innovation_code