Business Analyse Technieken: modelleren en realiseren van requirements

Pragmatische technieken voor de business analist | Module 2

14/06/2022

Christian Gijsels

NCOI LEARNING

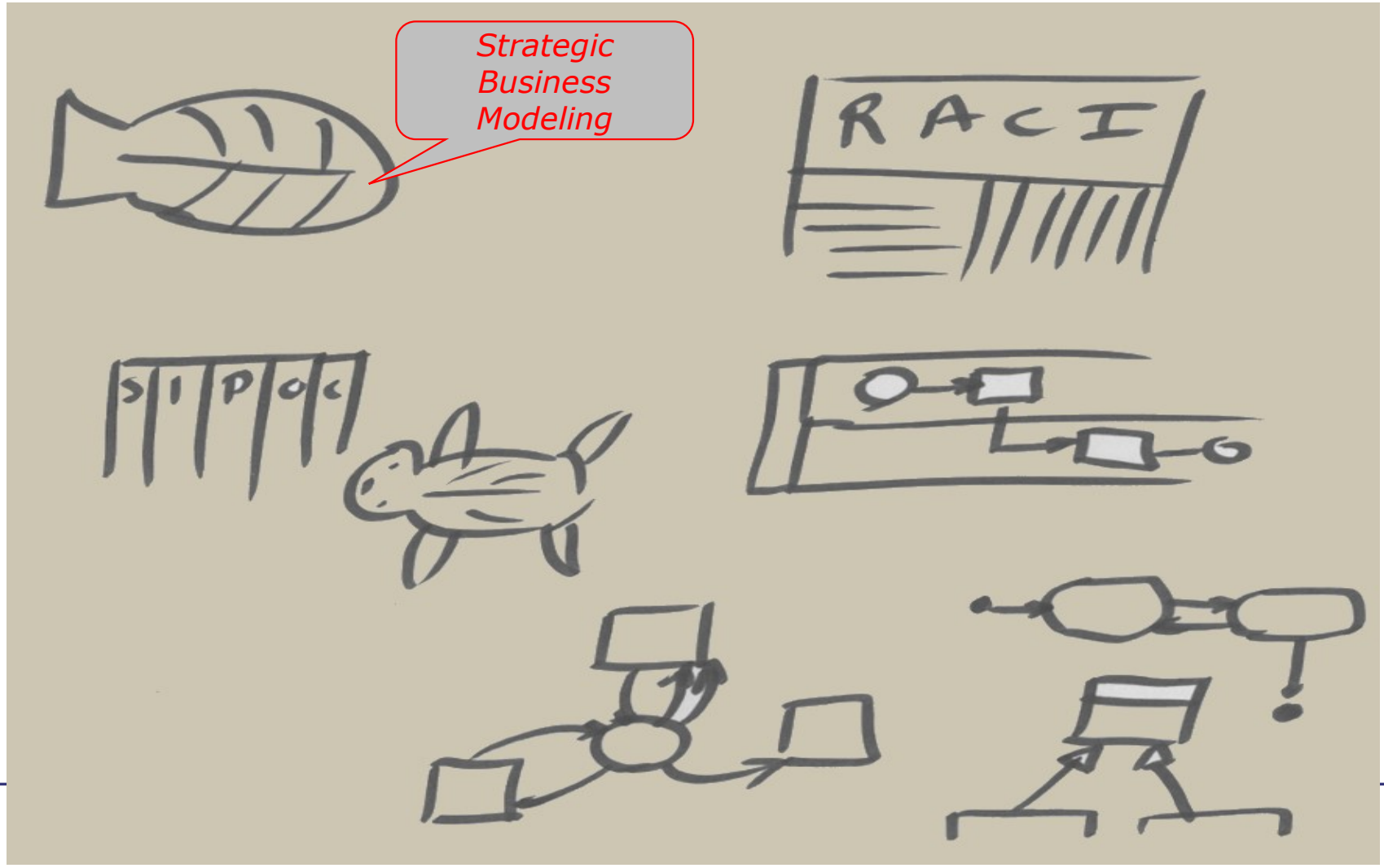
GIJSELSDOTCOM CONSULTING | the Partner for your Digital Transformation www.gijsels.com

Index

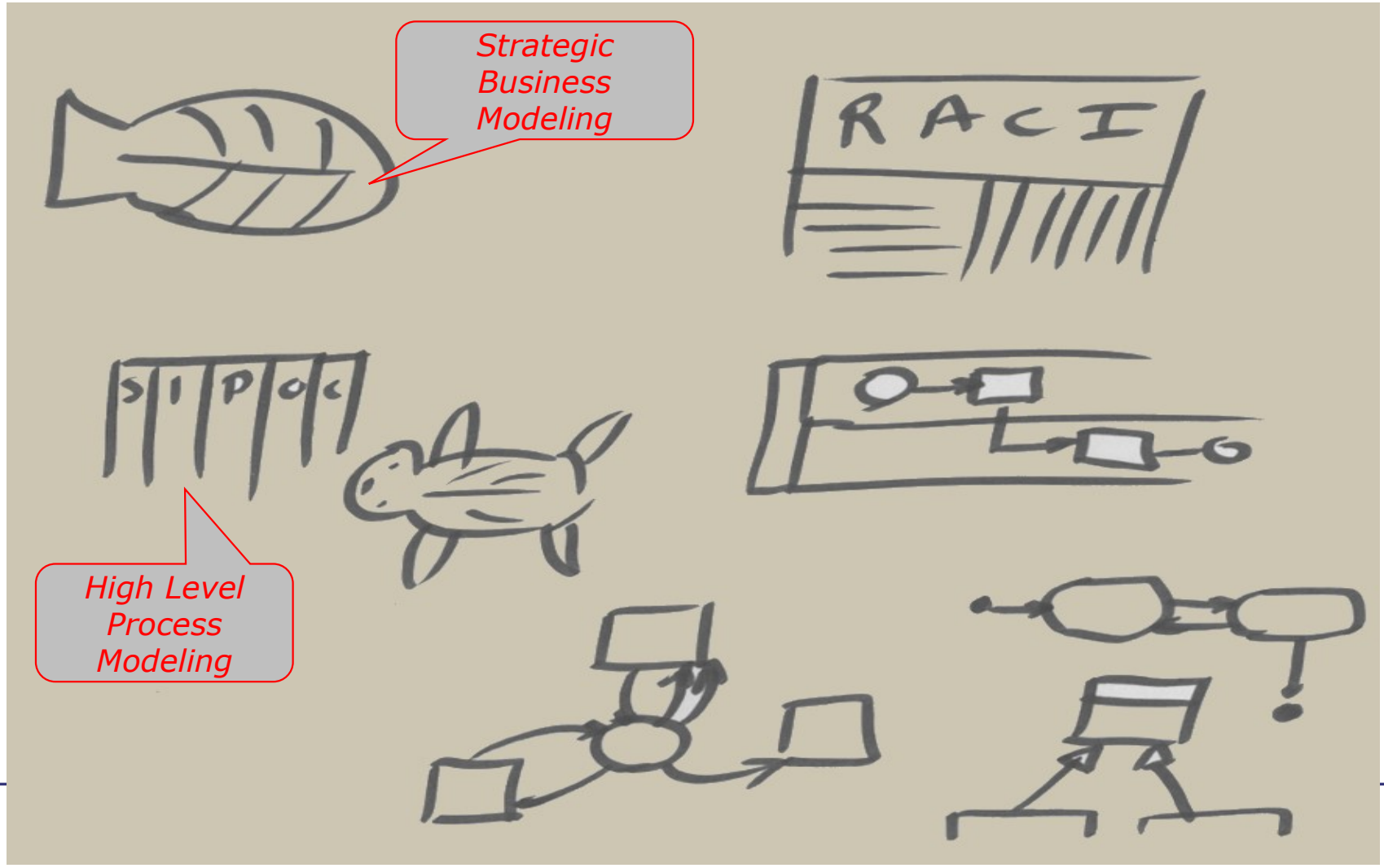
Part I: Positioning Business Analysis

POSITIONING BA TECHNIQUES

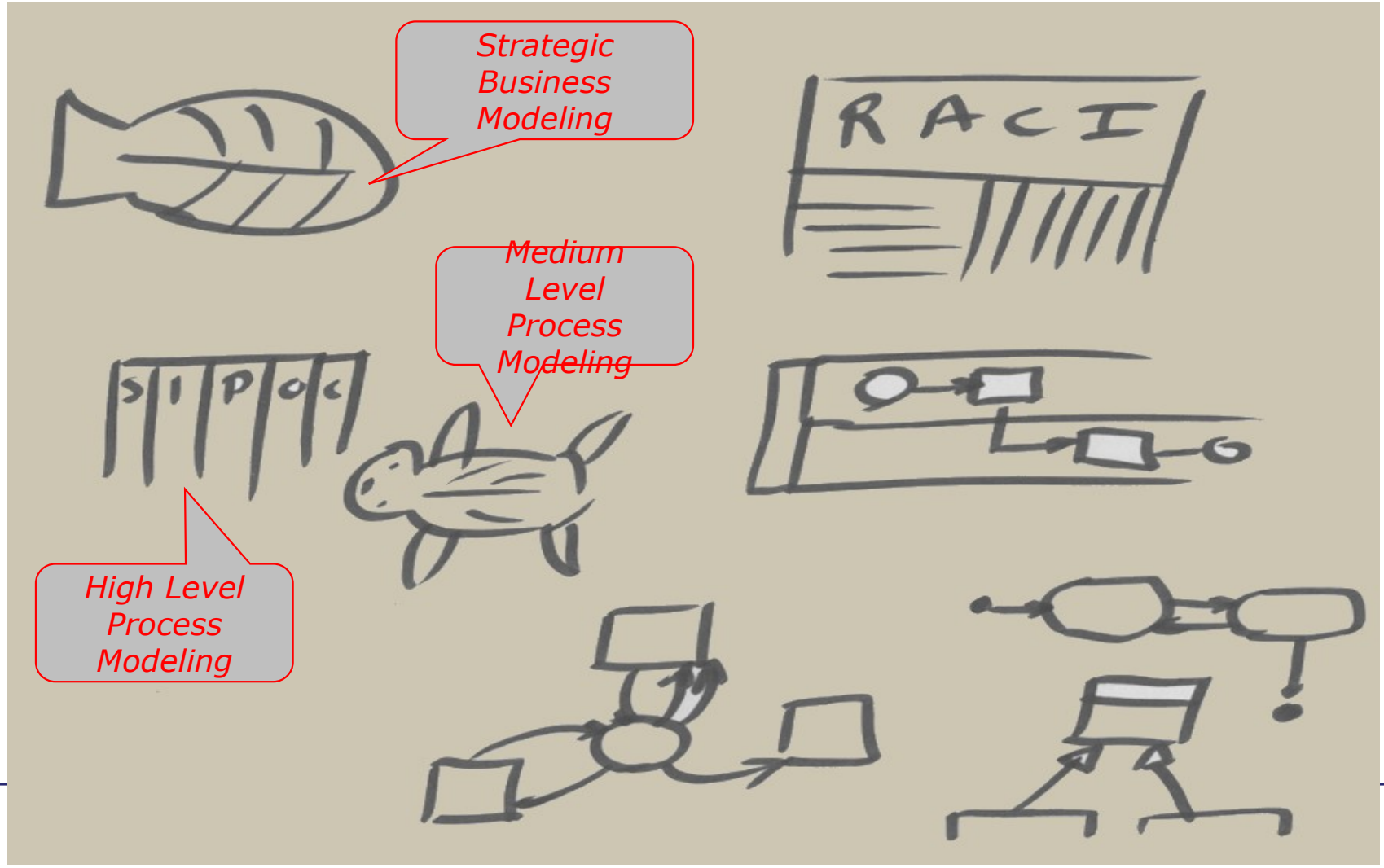
POSITIONING BA TECHNIQUES



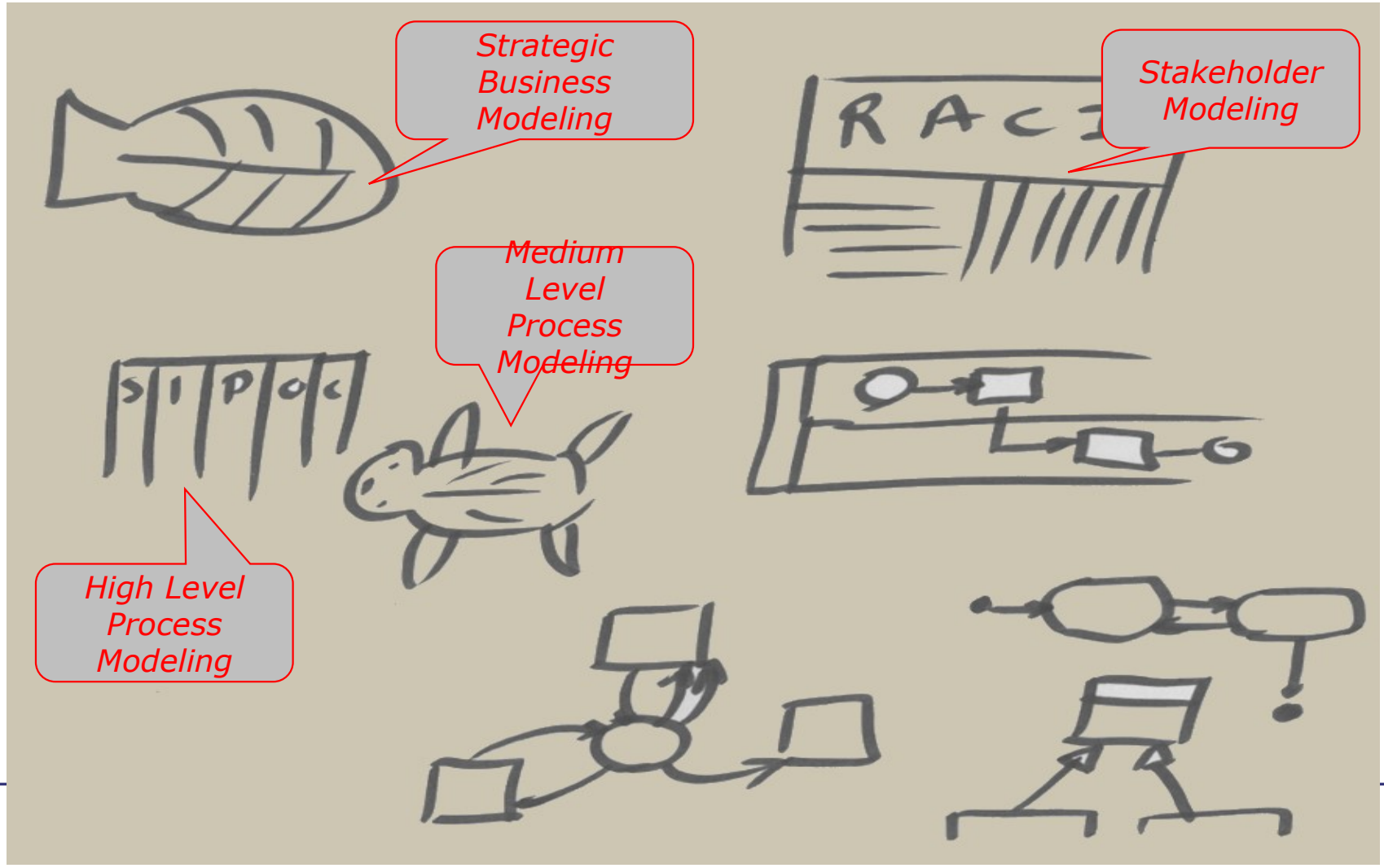
POSITIONING BA TECHNIQUES



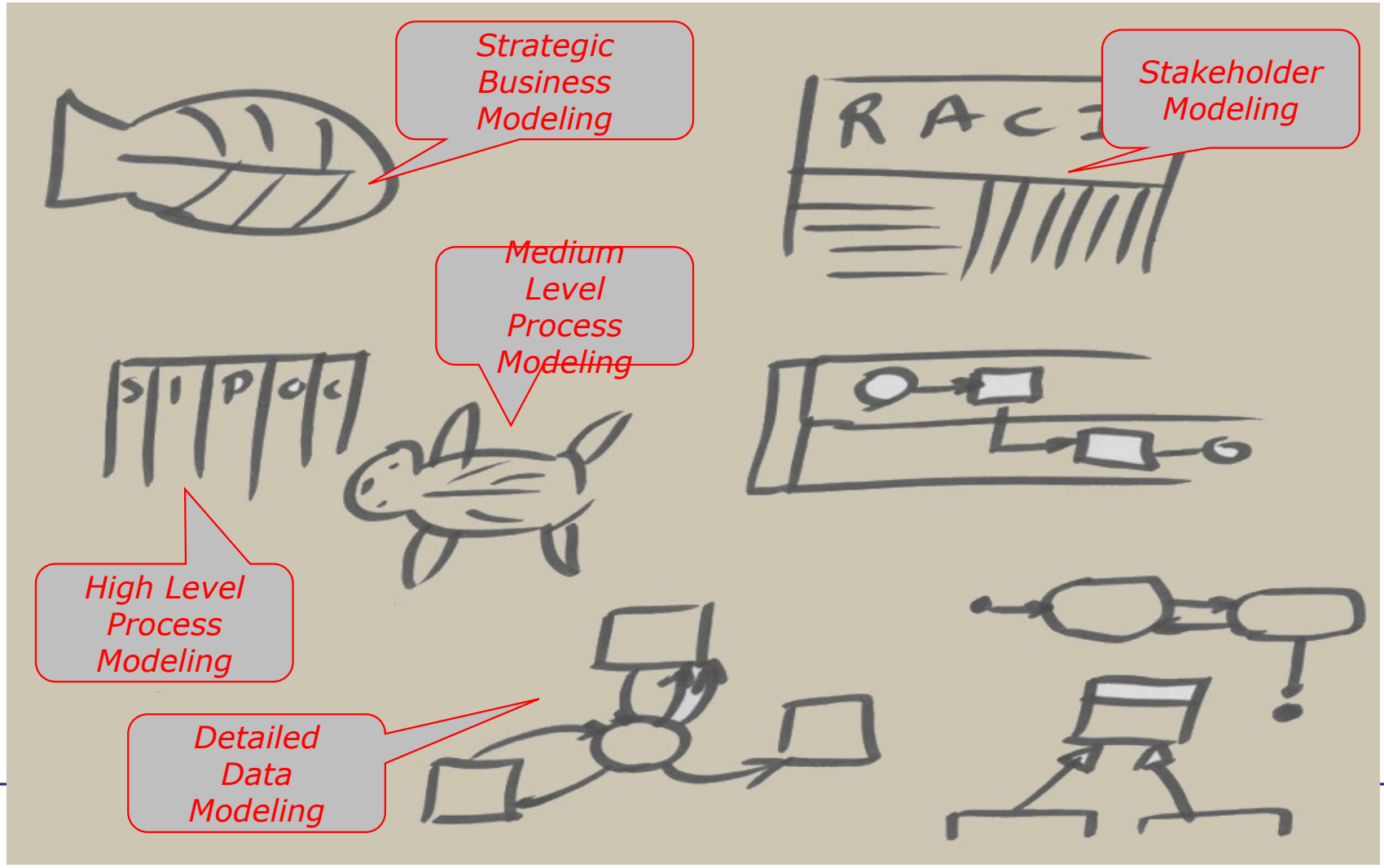
POSITIONING BA TECHNIQUES



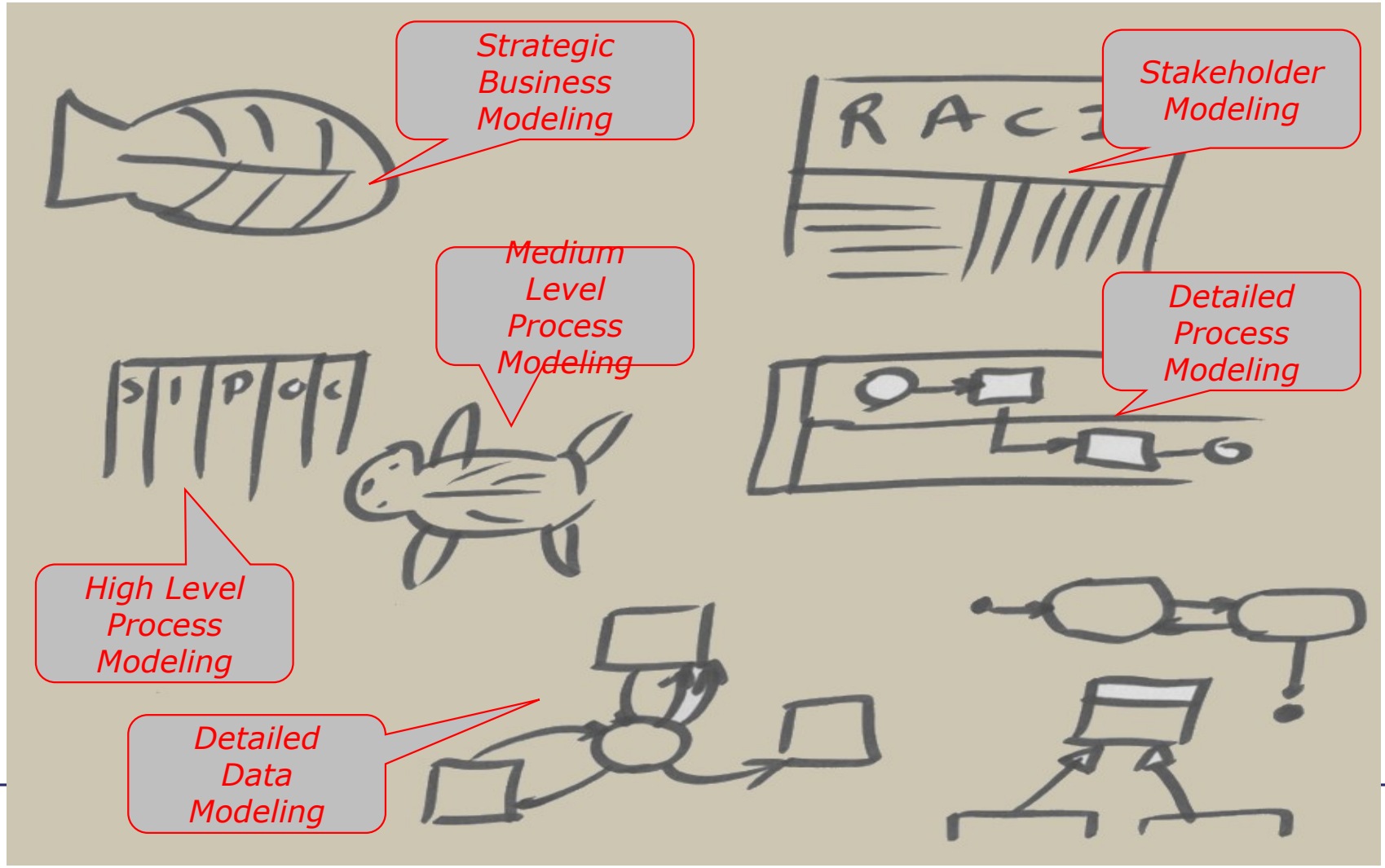
POSITIONING BA TECHNIQUES



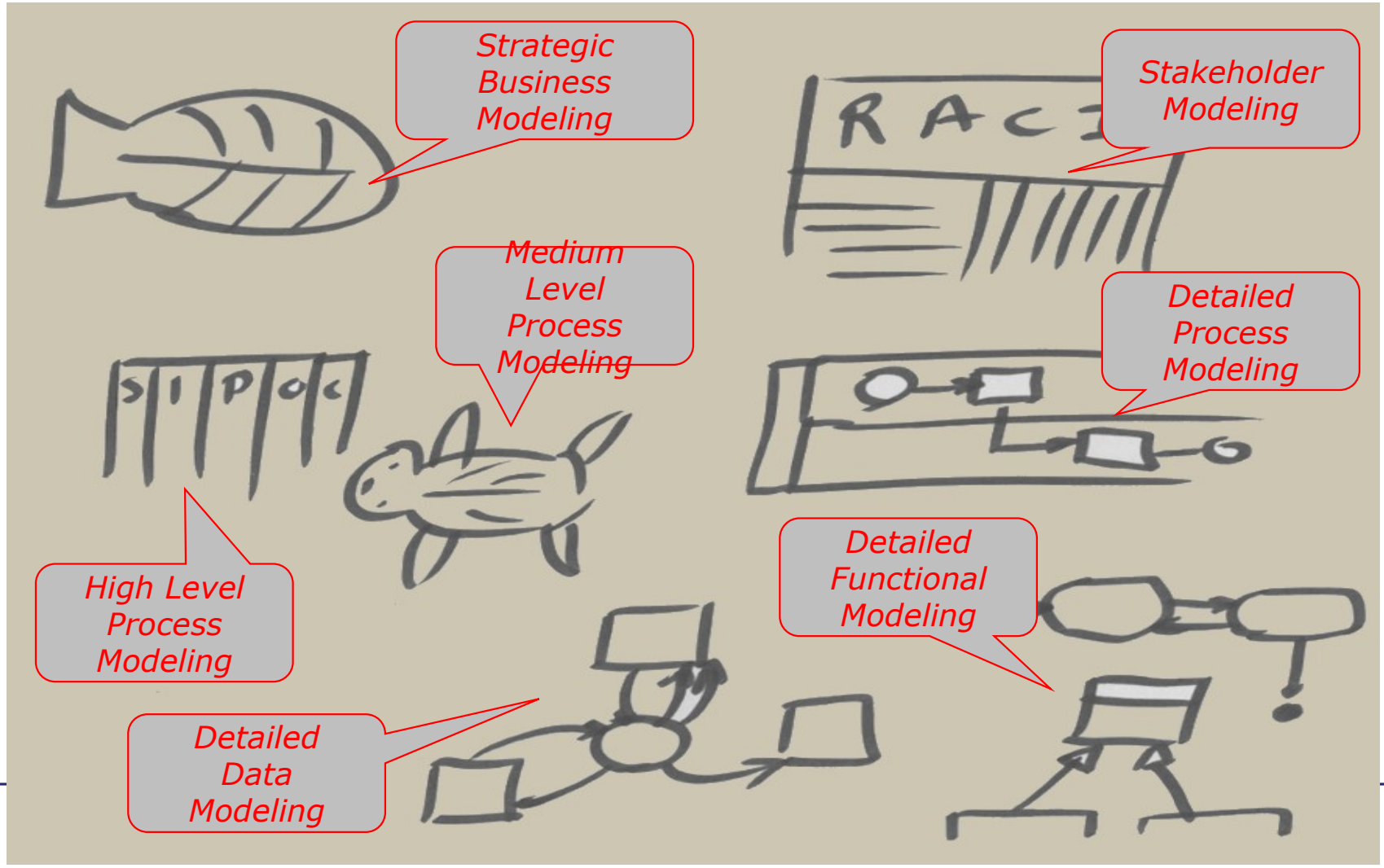
POSITIONING BA TECHNIQUES



POSITIONING BA TECHNIQUES



POSITIONING BA TECHNIQUES



Part II: Cause-and-effect

- Introduction
  - Practical
  - Questions
  - Predefined categories
- How to
  - 3 steps
- Example

- Demo
- Exercise

Cause-and-effect Diagrams - Introduction

Practical; what is it - Root Cause Analysis

- **Root Cause Analysis (RCA)** is a management process that **seeks to locate the ultimate cause(s)** behind performance or process-related **problems** in a business or engineering environment and then proceed to **resolve the problem by treating these underlying causes.**

- The **benefits of Root Cause Analysis**, as a result, are the deeper investigation into the reason for the occurrence in the first place. The root cause or causes might be much **deeper** than outward symptoms reveal, and several layers may have to be pushed aside to reach the "**root**" cause. So, the focus is on analysis of this fabled "root cause" that propagated forward and manifested in the form of the problem at hand, rather than exclusively treating the symptoms, as troubleshooting does.

Cause-and-effect Diagrams - Introduction

Practical; what is it - Root Cause Analysis

Summarized, the goals of Root Cause Analysis are:

- **Failure identification**: what exactly went wrong
- **Failure analysis**: why it happened, discover the root cause
- **Failure resolution**: provide a solution that prevents recurrence

Techniques for **Root Cause Analysis:**

- **Cause-and-effect diagrams**
- Fault tree
- Check sheets
- …

Cause-and-effect Diagrams - Introduction

Practical; what is it - Cause-and-effect diagram

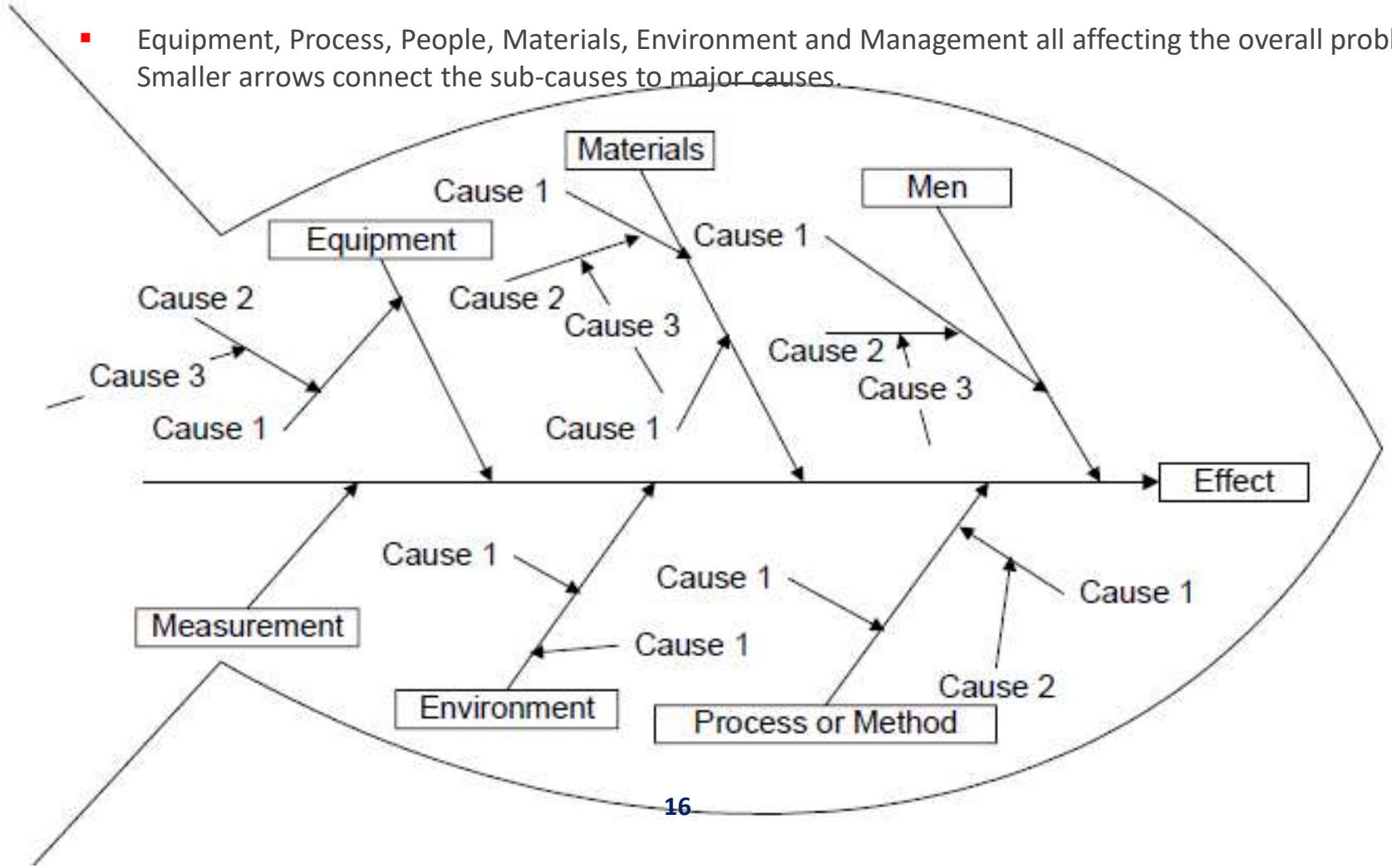
- Cause-and-effect diagrams (also called Ishikawa diagrams, fishbone diagrams, herringbone diagrams or Fishikawa) are causal diagrams that show the **causes of a specific event** - created by Kaoru Ishikawa (1968)

- Cause-and-effect diagrams, in fishbone shape, showing factors of Equipment, Process, People, Materials, Environment and Management, all affecting the overall problem. Smaller arrows connect the sub-causes to major causes.

Cause-and-effect Diagrams - Introduction

Practical; what is it - Cause-and-effect diagram

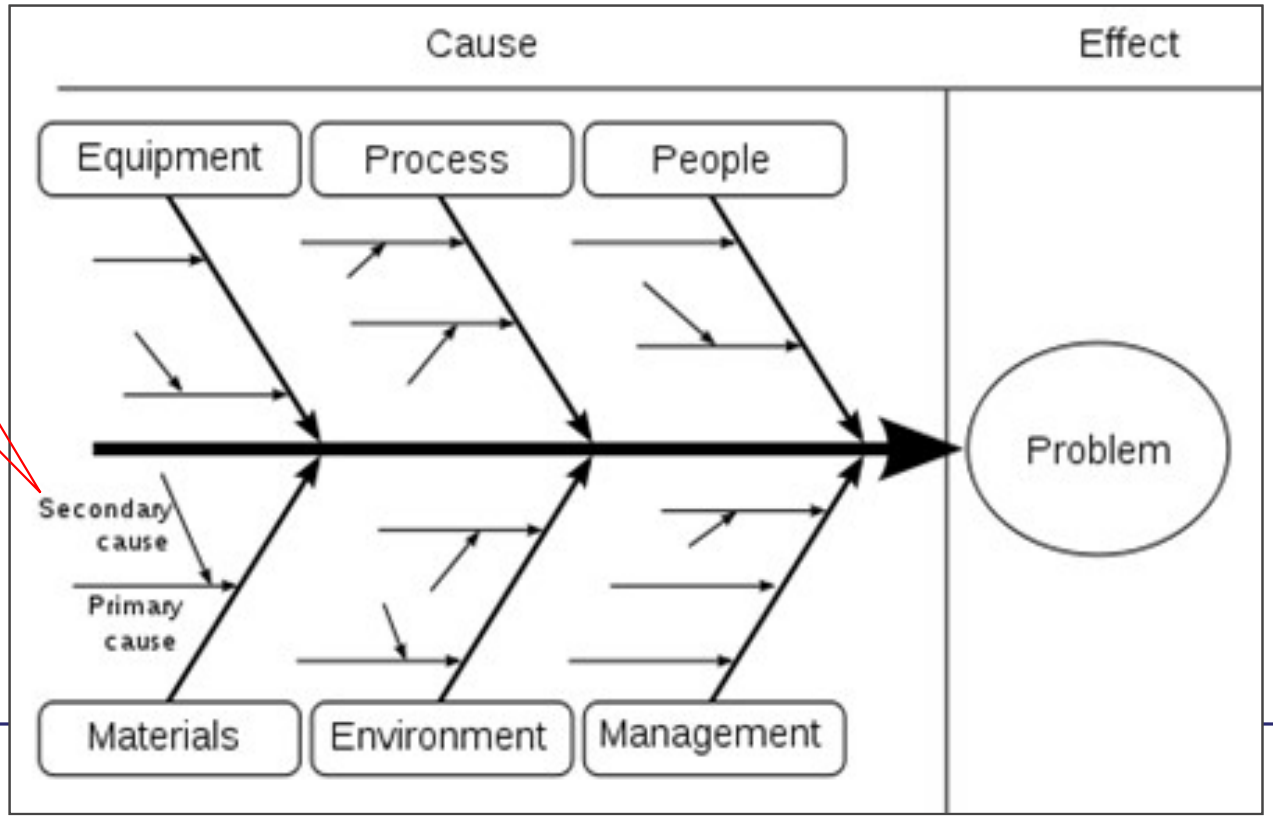
- Equipment, Process, People, Materials, Environment and Management all affecting the overall problem. Smaller arrows connect the sub-causes to major causes.



Cause-and-effect Diagrams - Introduction

Practical; what is it - Cause-and-effect diagram

The root cause or causes might be much **deeper** than outward symptoms reveal, and several layers may have to be pushed aside to reach the "**root**" cause.



Cause-and-effect Diagrams - Introduction

Questions to be asked while building a Cause-and-effect diagram

- **Machines** – Was the correct tool/tooling used? - Does it meet production requirements? - Does it meet process capabilities?
- **Material** – Is all needed information available and accurate? – Can information be verified or cross-checked?
- **Method** – Were the workers trained properly in the procedure? – Was the testing performed statistically significant? – Was data tested for true root cause?
- **Environment** – Is the process affected by temperature changes over the course of a day? – Is the process affected by humidity, vibration, noise, lighting, etc.?
- …

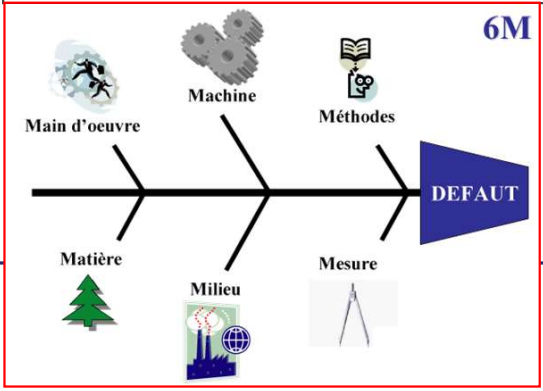
Cause-and-effect Diagrams - Predefined categories

Predefined categories - The 6 M's – 5 S's – 7 P's

- Causes in the diagram are often categorized, such as to the 6 M's, described below. Cause-and-effect diagrams can reveal key relationships among various variables, and the possible causes provide additional insight into process behavior.

The 6 M's (used in manufacturing)

- Machine (technology)
- Method (process)
- Material (Includes Raw Material, Consumables and Information.)
- Man Power (physical work)/Mind Power (brain work)
- Measurement (Inspection)
- Milieu/Mother Nature (Environment)



The 5 S's (used in service industry)

- Surroundings
- Suppliers
- Systems
- Skills
- Safety

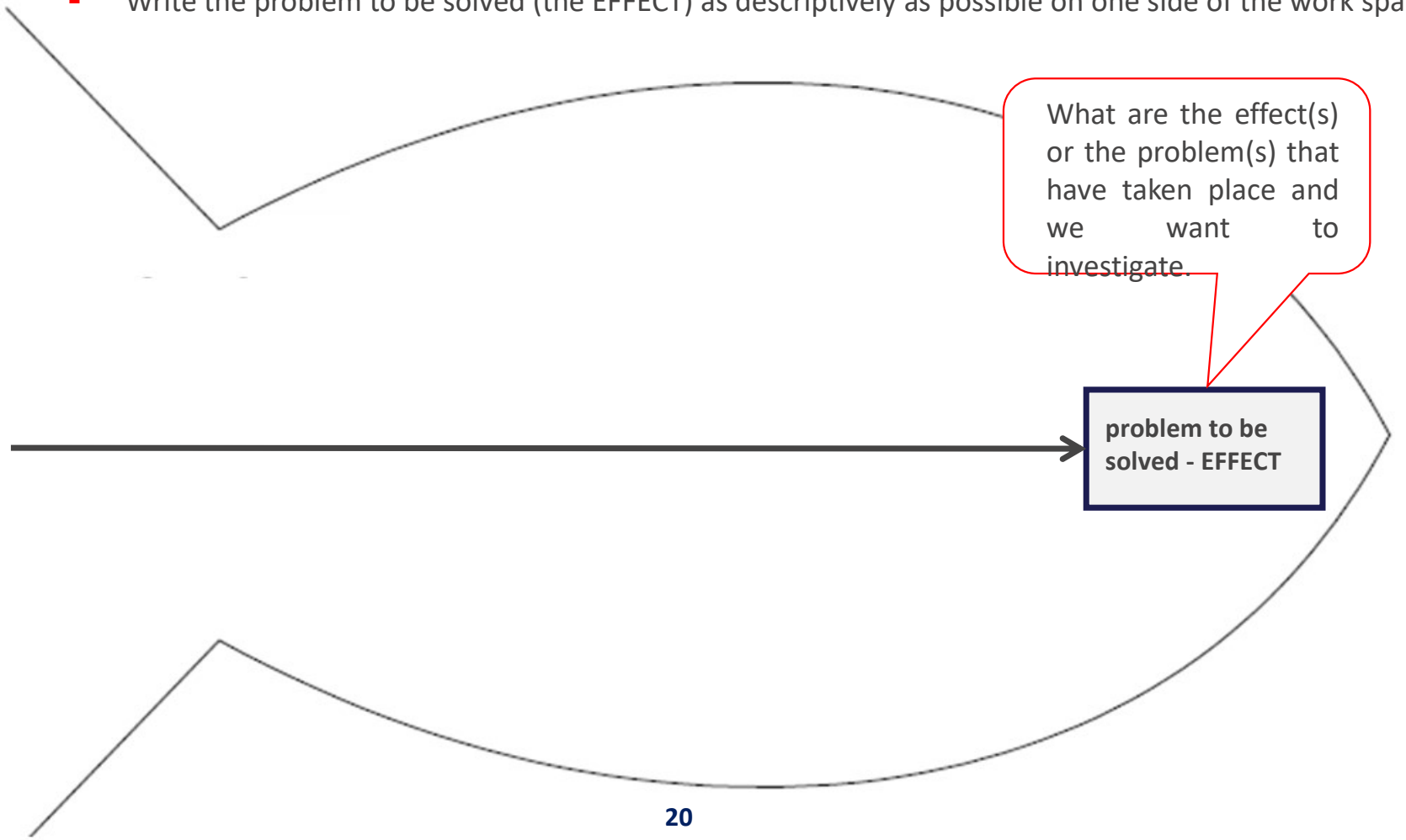
The 7 P's (used in manufacturing industry)

- Product=Service
- Price
- Place
- Promotion
- People/personnel
- Process
- Physical Evidence

Cause-and-effect Diagrams - How

Make a Cause-and-effect diagram

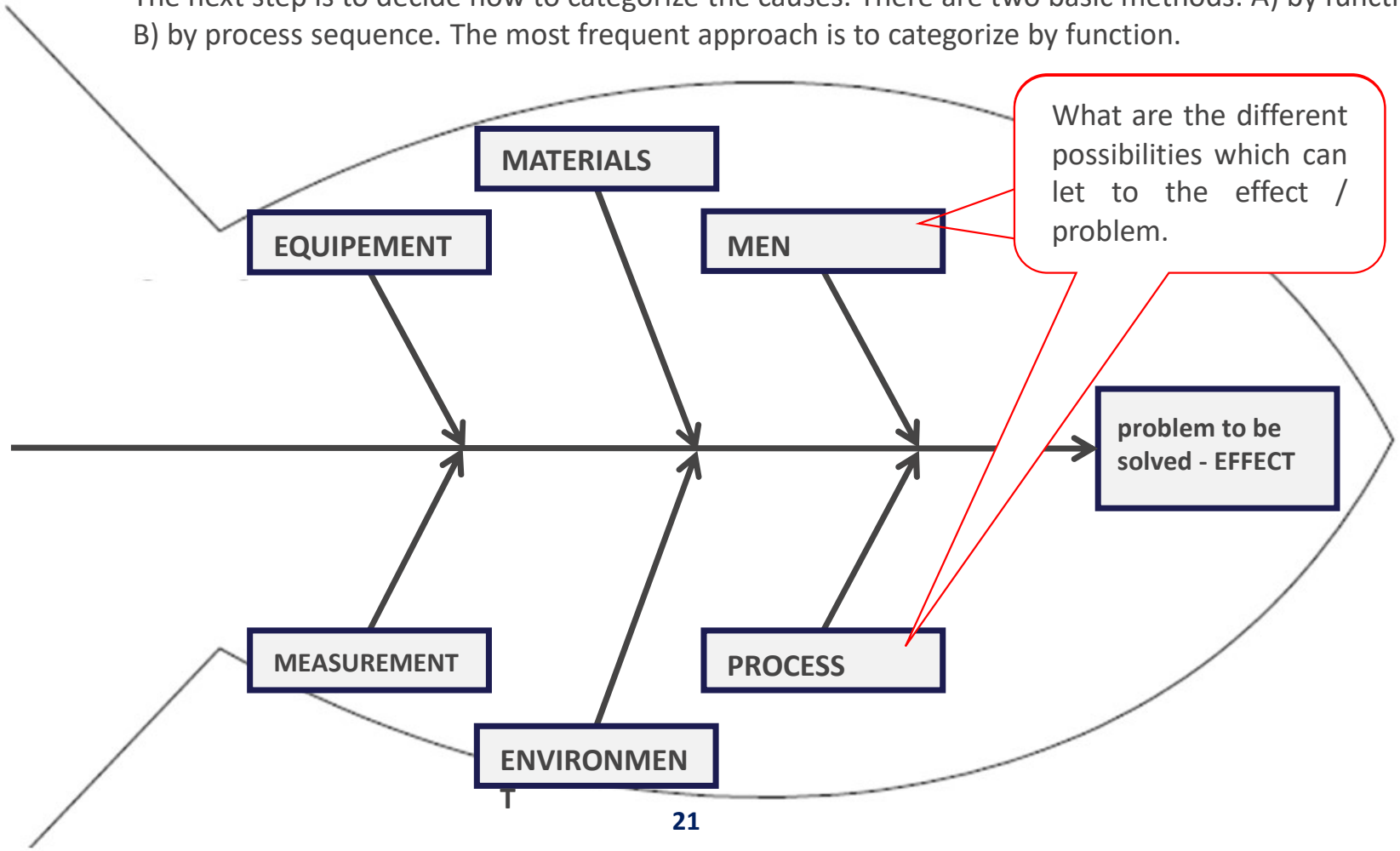
- Write the problem to be solved (the EFFECT) as descriptively as possible on one side of the work space.



Cause-and-effect Diagrams - How

Make a Cause-and-effect diagram

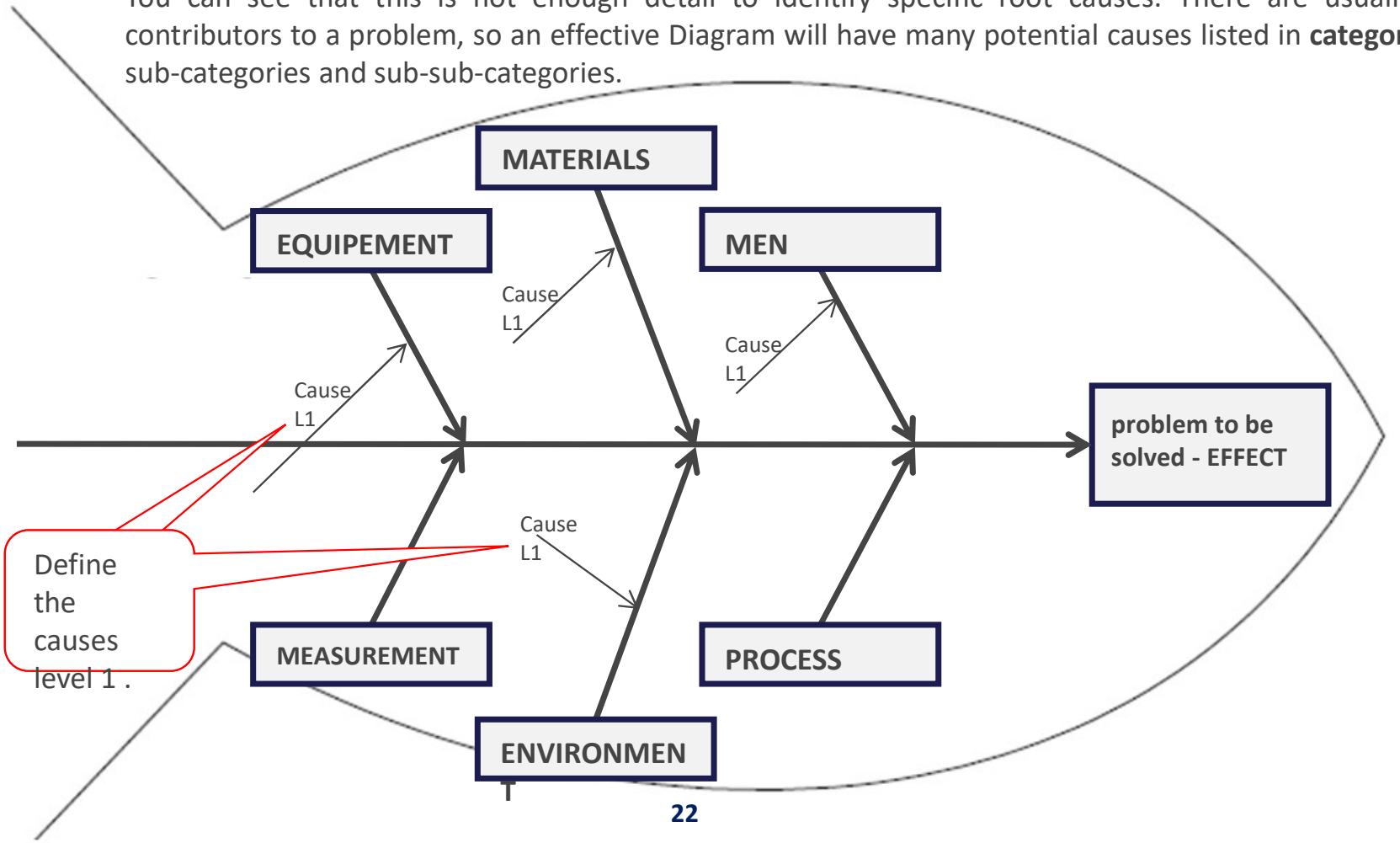
- The next step is to decide how to categorize the causes. There are two basic methods: A) by function, or B) by process sequence. The most frequent approach is to categorize by function.



Cause-and-effect Diagrams - How

Make a Cause-and-effect diagram

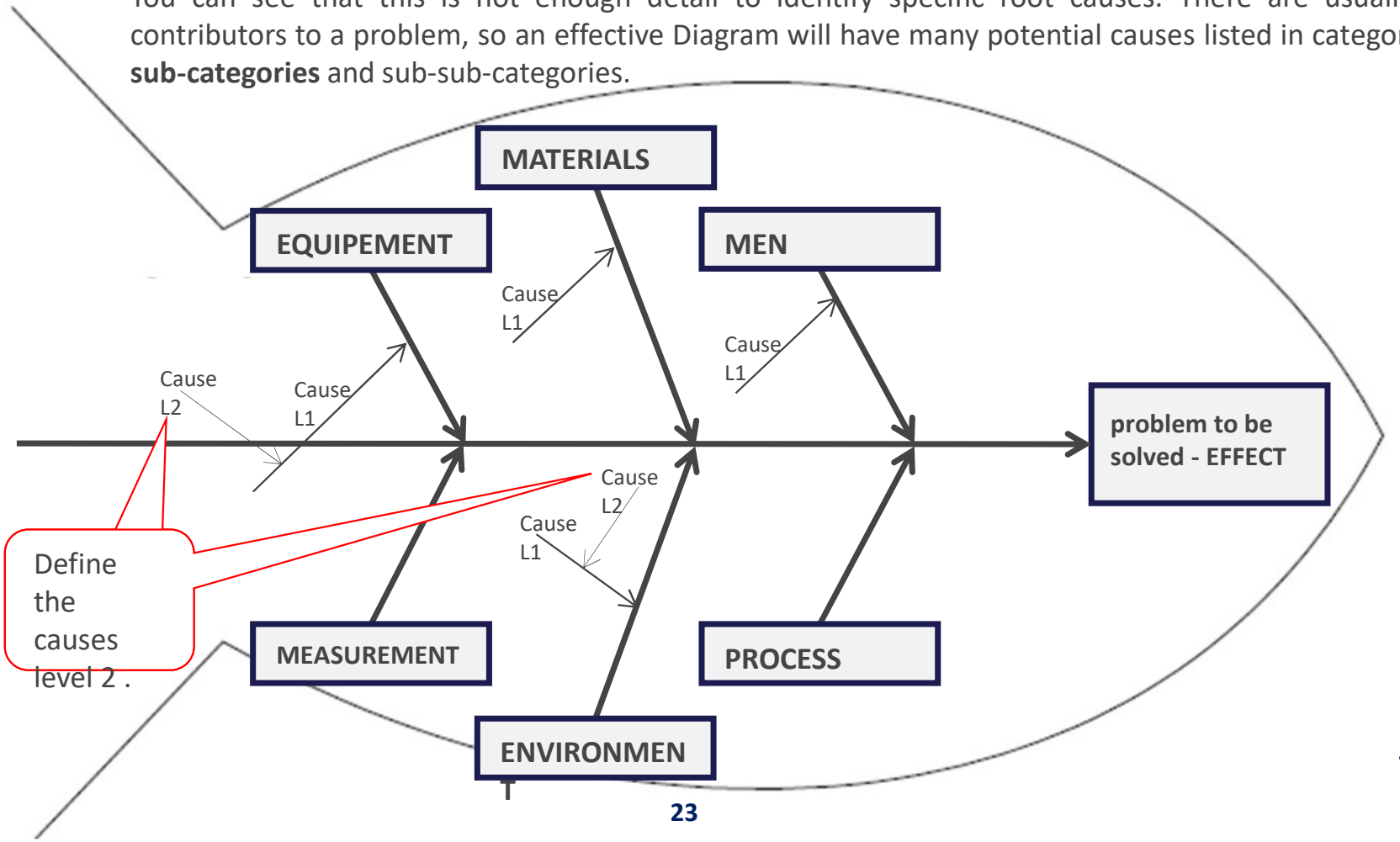
- You can see that this is not enough detail to identify specific root causes. There are usually many contributors to a problem, so an effective Diagram will have many potential causes listed in **categories** and sub-categories and sub-sub-categories.



Cause-and-effect Diagrams - How

Make a Cause-and-effect diagram

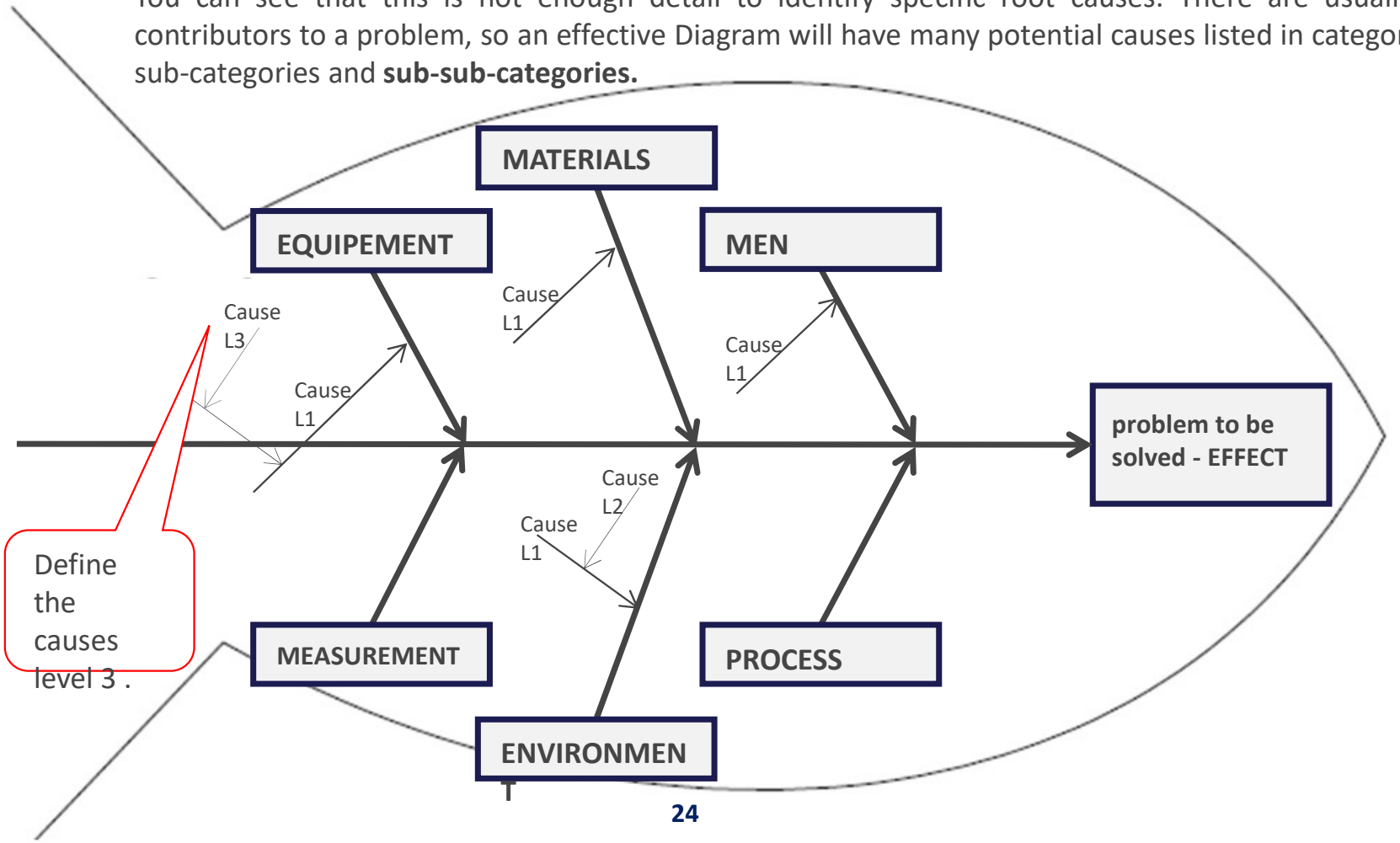
- You can see that this is not enough detail to identify specific root causes. There are usually many contributors to a problem, so an effective Diagram will have many potential causes listed in categories and **sub-categories** and sub-sub-categories.



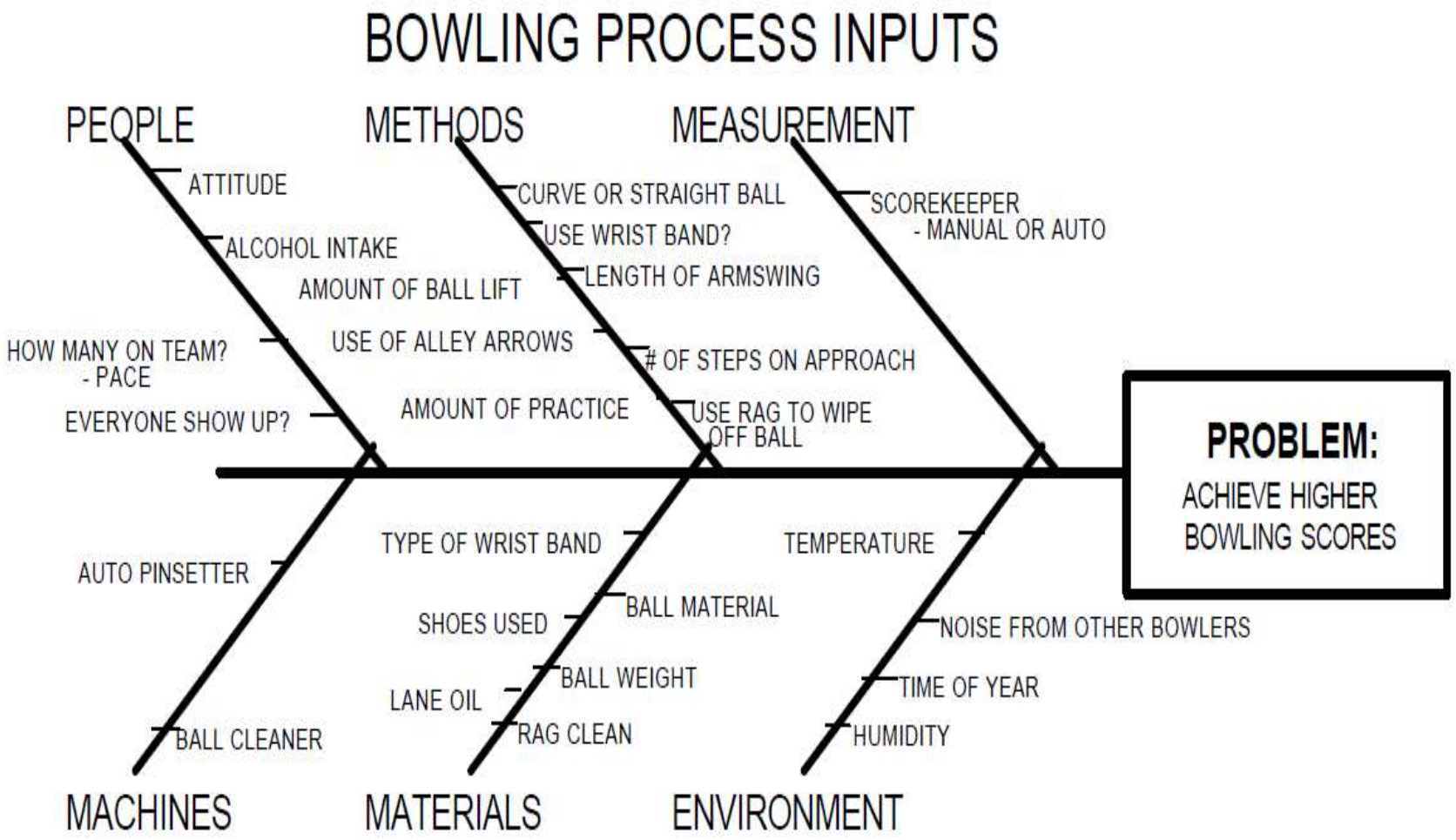
Cause-and-effect Diagrams - How

Make a Cause-and-effect diagram

- You can see that this is not enough detail to identify specific root causes. There are usually many contributors to a problem, so an effective Diagram will have many potential causes listed in categories and sub-categories and **sub-sub-categories.**

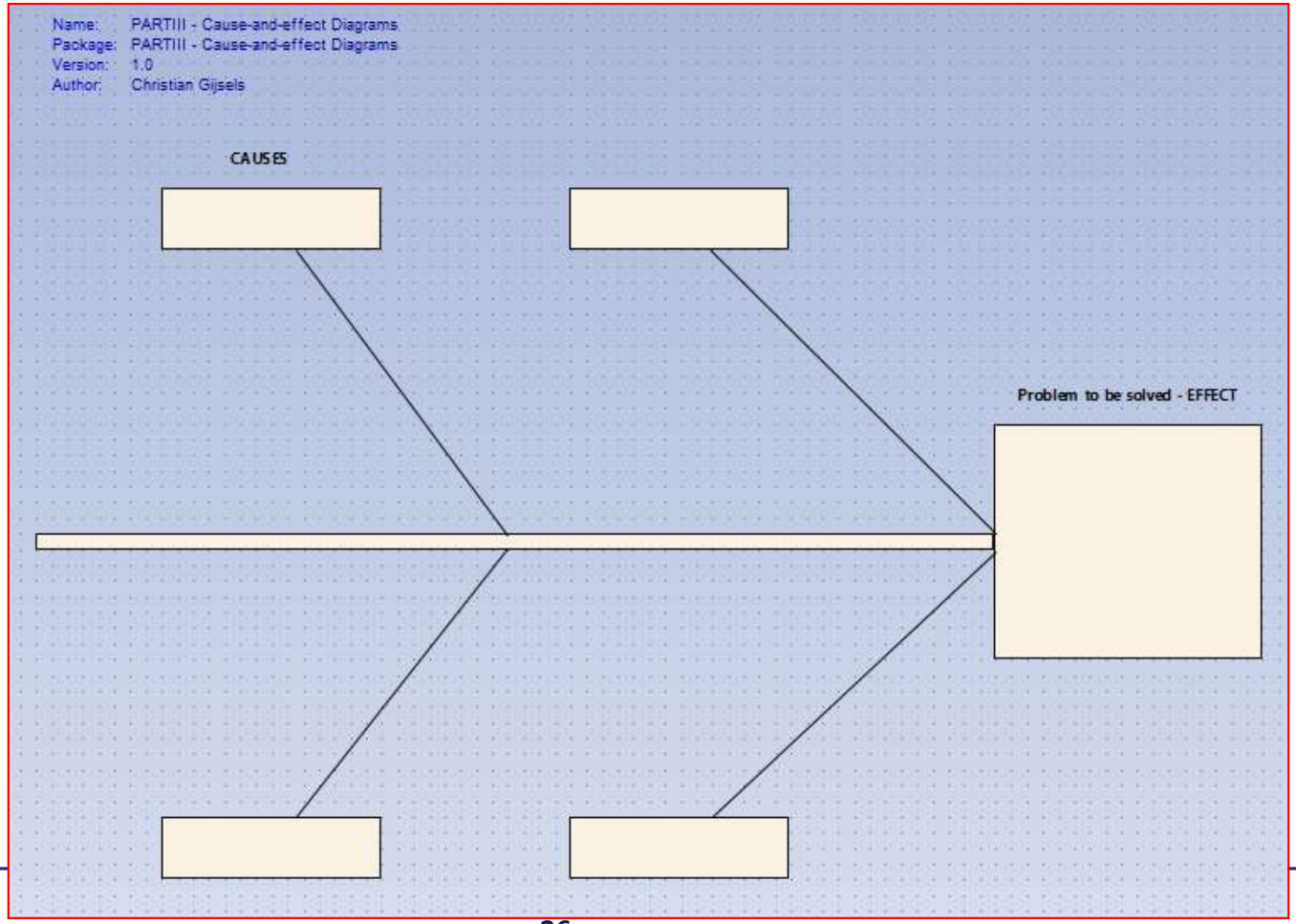


Cause-and-effect Diagrams - Example

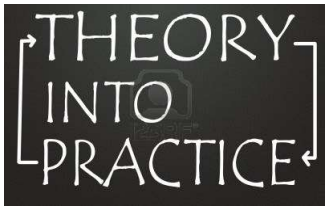


Cause-and-effect Diagrams - Demo

Demo with Enterprise Architect



Cause-and-effect Diagrams - Exercise

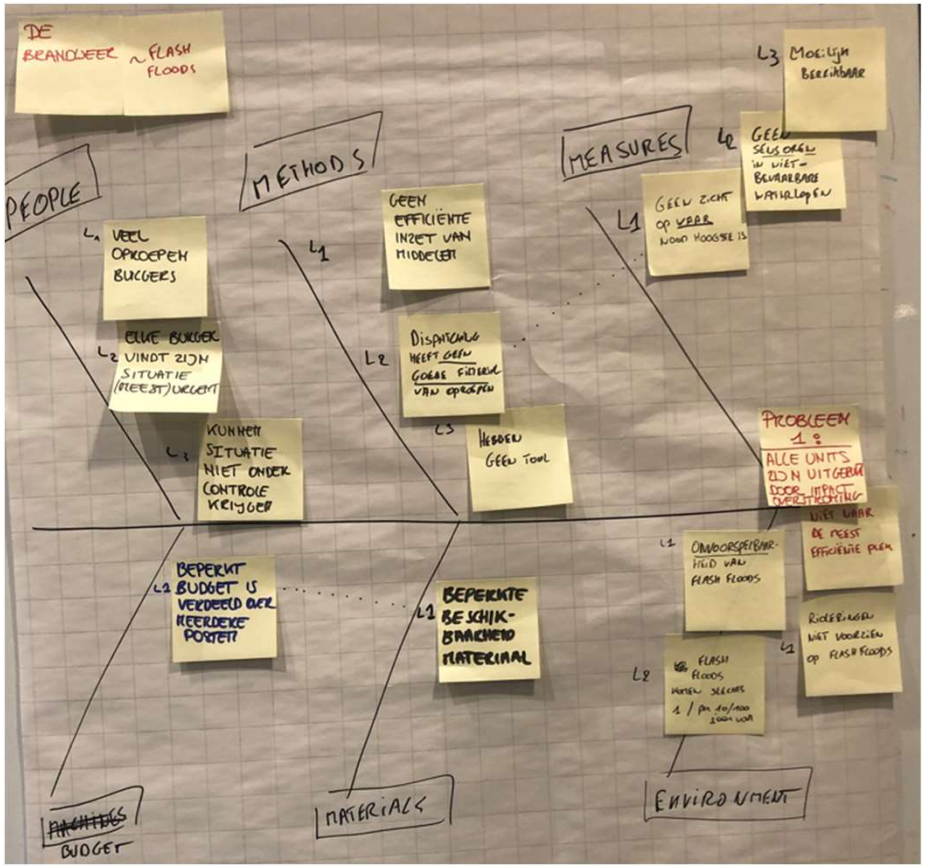


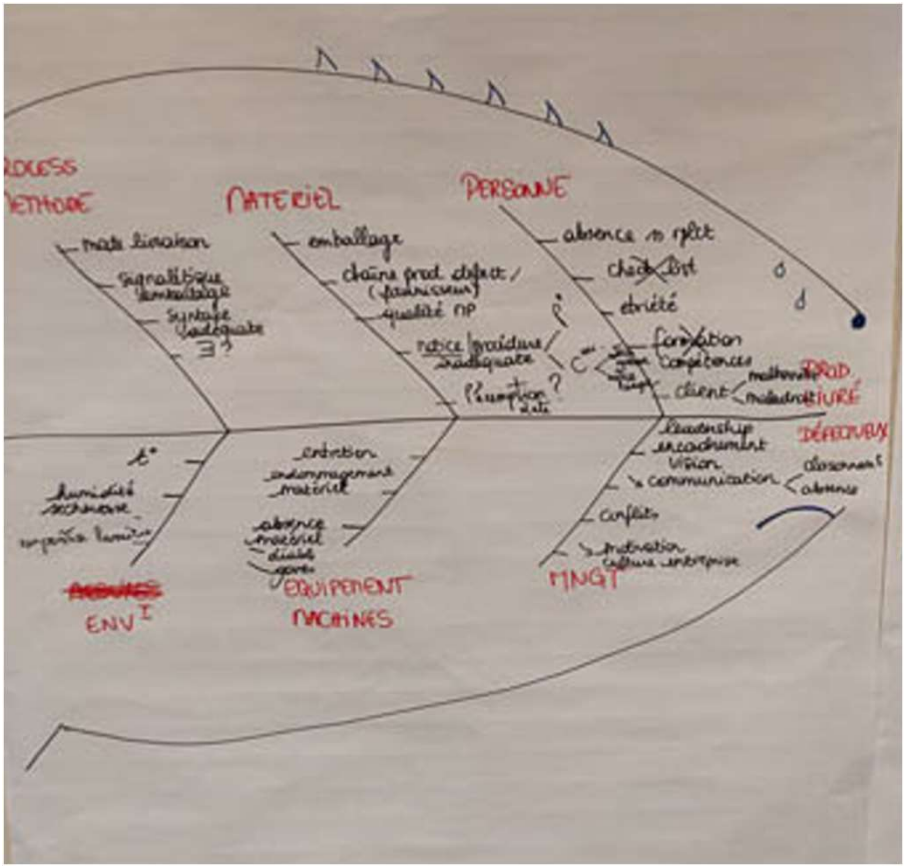
Exercise: Make a Cause-and-effect Diagram

- Make a Cause-and-effect Diagram of a Defect Flashlight

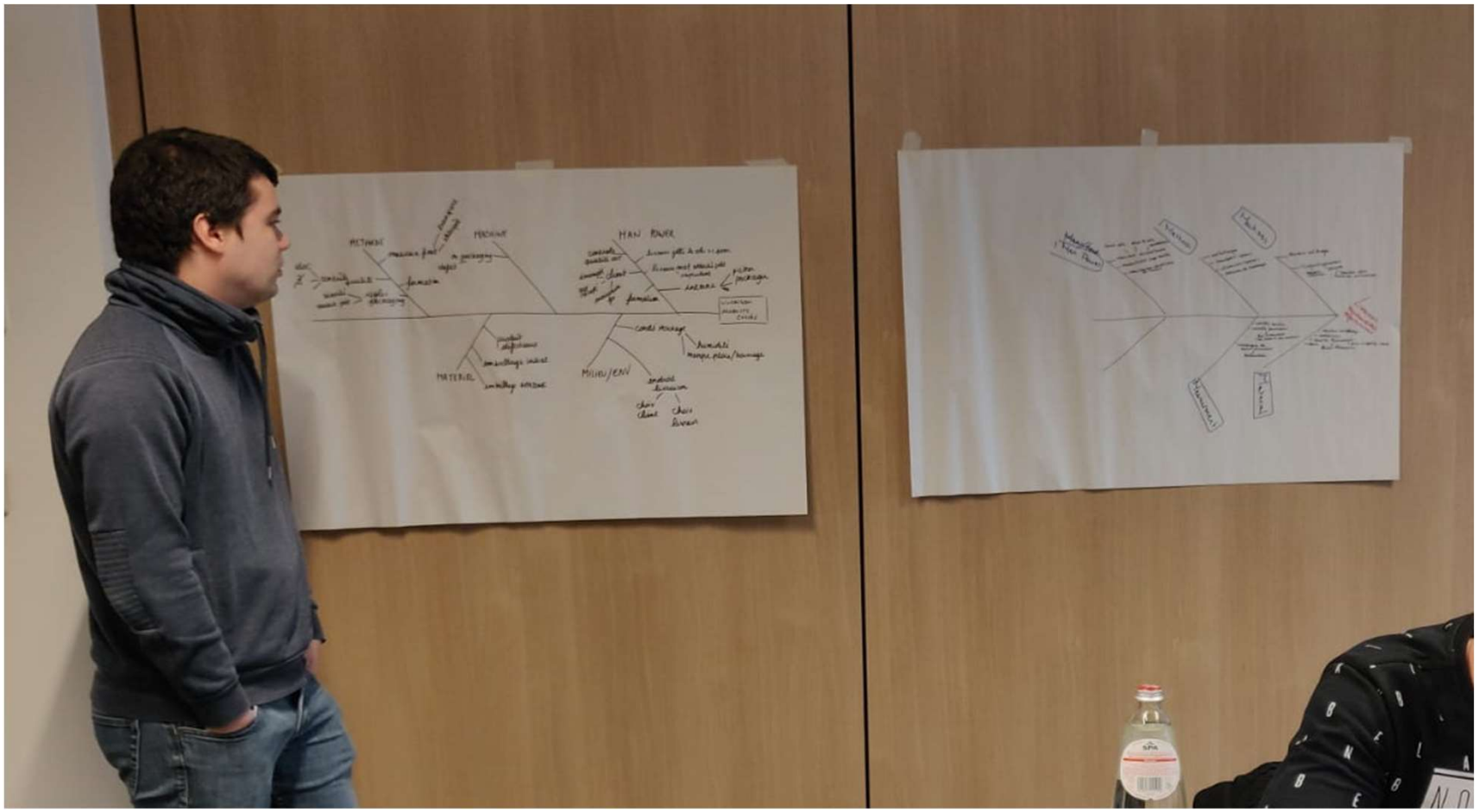
- Present your Cause-and-effect Diagram

Cause-and-effect Diagrams - Examples





Cause-and-effect Diagrams - Examples



Part III: SIPOC Diagram

- Positioning
- What is a SIPOC diagram
- Traditional SIPOC diagram layout
- The steps
- Examples

- Demo
- Exercise

SIPOC Diagrams - Introduction

What is a SIPOC diagram

SIPOC is structured high level description of the current process, with:

- **S** = **Supplier** of inputs of the process
  (Individuals, departments, organisations, …)
- **I** = **Inputs**
  (information, material, …)
- **P** = **Process** itself explained in a few blocks with clear beginning and end step
  (the macro steps)
- **O** = **Output** of this process (can be from a specific step of at the end)
  (products, services, …)
- **C** = Who is the **Customer**, uses any of the outputs?

SIPOC Diagrams - Introduction

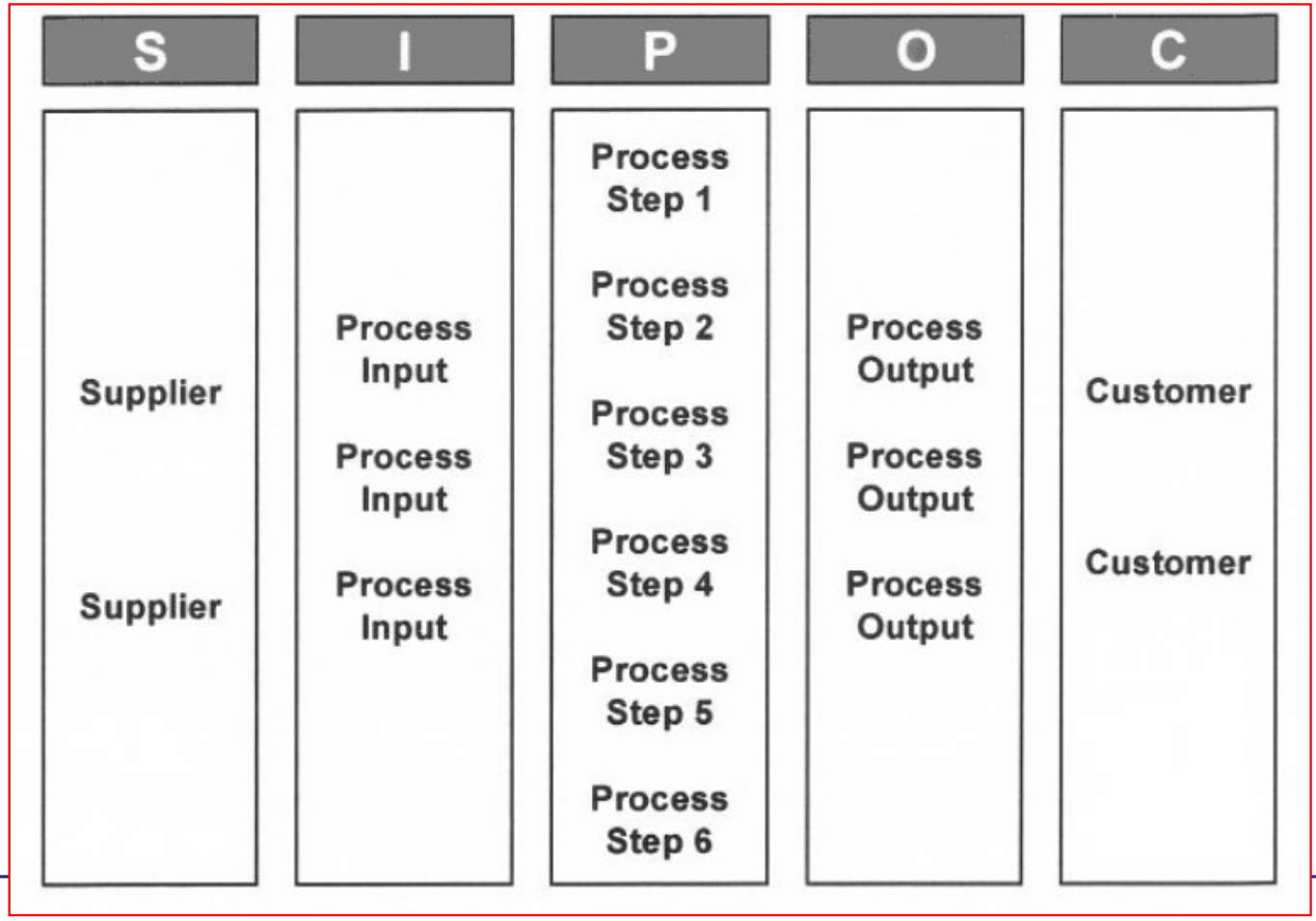
What is a SIPOC diagram

The **SIPOC**:

- Allows everyone on the team to have the **same picture of the process**
- Helps you clarify identify the **beginning** and **ending point** in your process that you will be focusing on
- Helps you understand the **major suppliers**, **inputs**, **outputs** and **customer** to keep focus
- Is a good **communication tool** when presenting your projects to others
- Assists the team in **keeping focused** and not going out of scope

SIPOC Diagrams - Layout

Traditional SIPOC diagram layout



SIPOC Diagrams – The steps

The steps 1 .. 3

Follow these steps to create your **SIPOC**:

- **Step 1:**
  List 5-8 **Major steps** in the process between the start and stop point (scope) of your project:
  >> Use action verbs
  >> List steps sequentially

- **Step 2:**
  List **Outputs** of your process:
  >> Focus on outputs of the whole process (not individual steps)
  >> Use nouns
  >> Consider outputs for both internal and external customers

- **Step 3:**
  List (internal and external) **Customers** of the outputs from your process:
  >> Co-workers, in same or different departments
  >> Consider systems
  >> Could also be an external company (e.g. vendors)

SIPOC Diagrams – The steps

The steps 4 .. 5

Follow these steps to create your **SIPOC**:

- **Step 4:**
  List **Inputs** to your process:
  >> Things that go into the process (physical objects, information, factors that influence the process)
  >> Use nouns
  >> Consider inputs from both internal and external suppliers

- **Step 5:**
  List (internal and external) **suppliers** of the inputs to your process:
  >> Co-workers, in same or different departments
  >> Consider systems input
  >> Vendors, customers (they can be suppliers to your process!)

Remark:
Different approaches are possible

SIPOC Diagrams - Template

Example: SIPOC Process Definition Template

Process: **Purpose:** **Owner:**				
Suppliers	**Inputs**	**Process Steps**	**Outputs**	**Customers**
Boundaries:				
Start-point:		End-point:		
Includes:		Excludes:		

SIPOC Diagrams - Examples

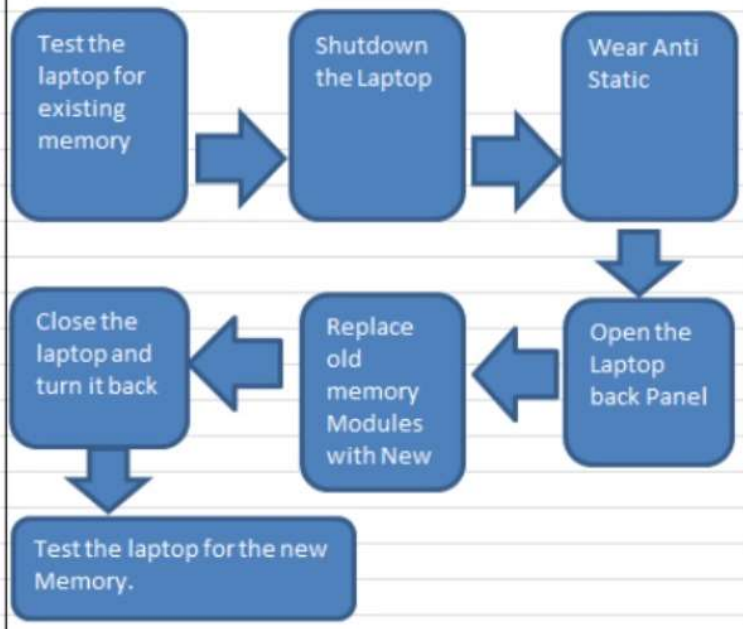
Example: Process: Recruit Staff

Process: Recruit Staff				
Suppliers	**Inputs**	**Process**	**Outputs**	**Customers**
Line Manager	Request to fill a vacancy	1. Specify needs 2. Authorise recruitment 3. Place adverts 4. Assess applicants 5. Offer appointment 6. Confirm start	New member of staff	Line Manager

SIPOC Diagrams - Examples

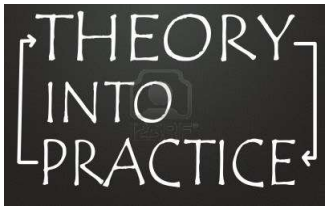
Example: Process: Upgrading Laptop Memory

Suppliers	Inputs	Process	Output	Customers
Mechanic	Laptop		Upgraded Laptop	Owner Of Laptop
Memory/Tools vendor. Example: Radioshak	Anti Static Gloves/Tools		Old Memory Modules	
Electricity Utility/Provider	Memory			
	Power Supply			
	Toolbox			



SIPOC Diagram for the example process of "Upgrading Laptop Memory"

SIPOC Diagrams - Exercise

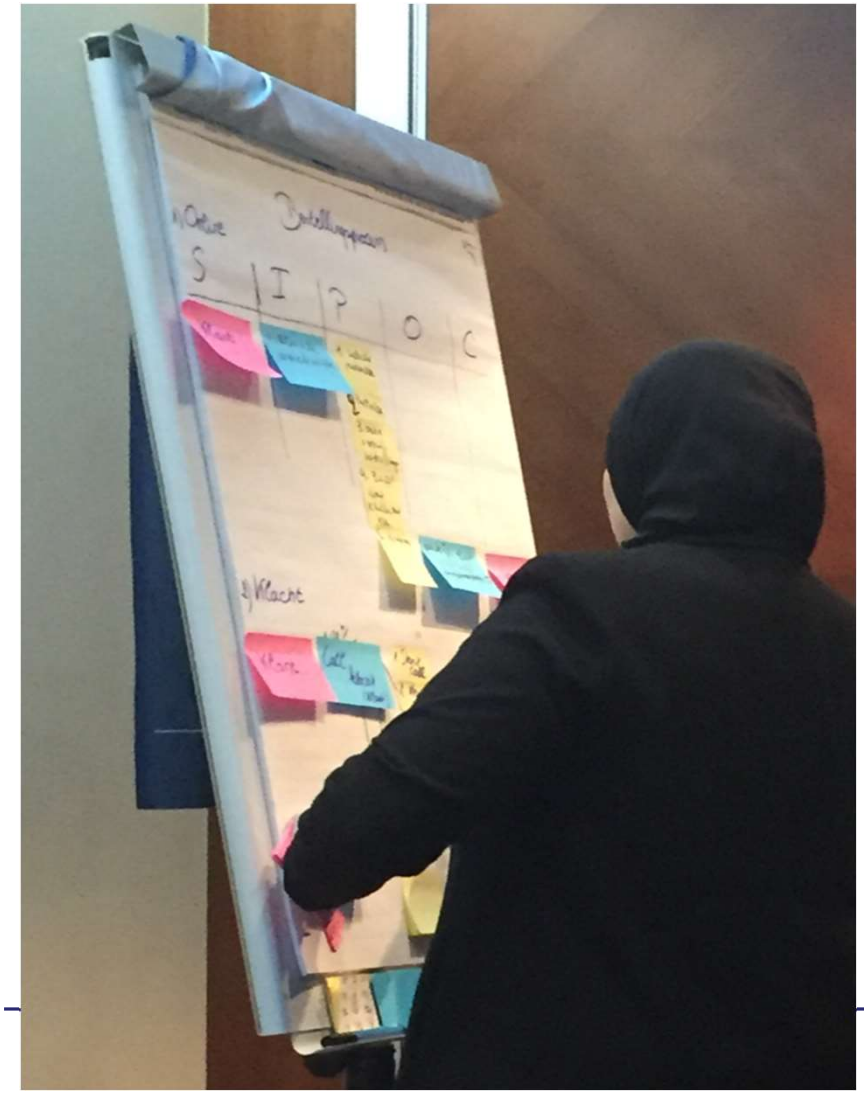


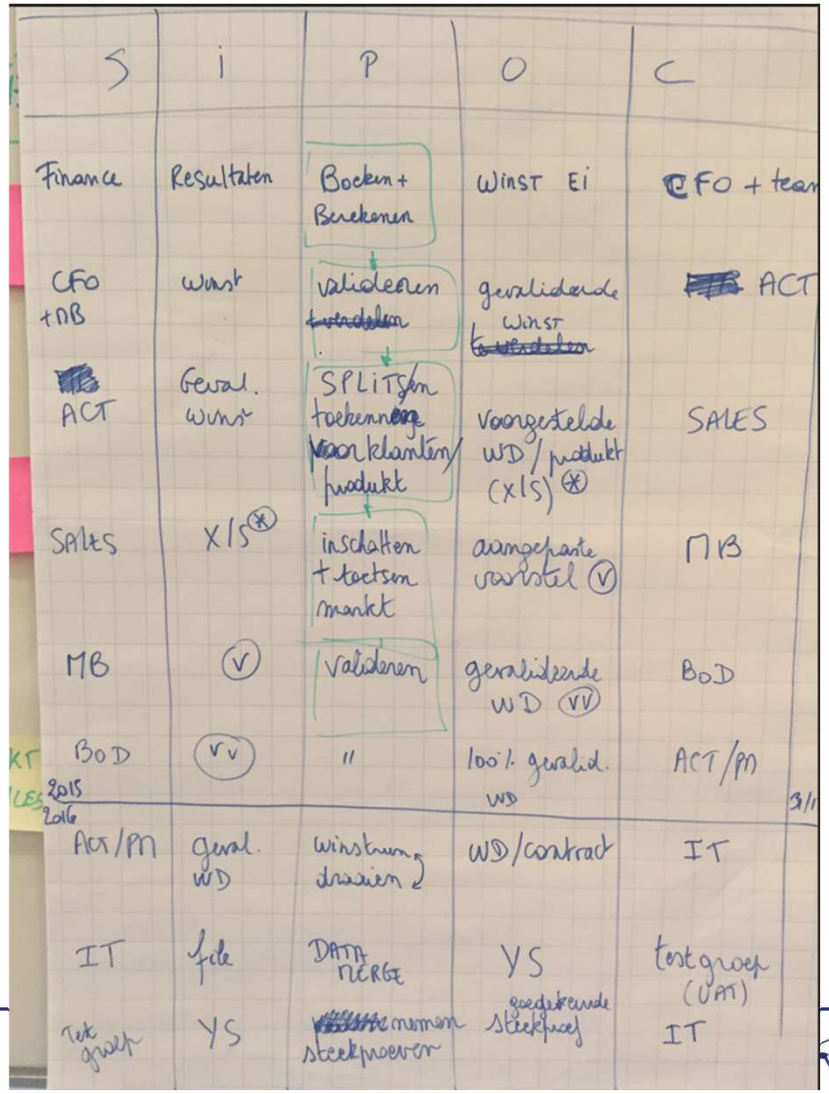
Exercise: Make a SIPOC Diagram

- Make a SIPOC Diagram of a Pizza Process

- Present your SIPOC Diagram

SIPOC Diagrams - Examples





Part IV: Turtle Diagram

- Positioning
- What is a turtle diagram
- The steps
- Examples

- Demo
- Exercise

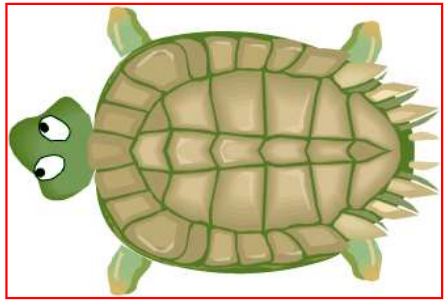
Positioning

Introduction

When all processes have been identified (as operational, support or management process) they should be analyzed taking the associated risks into consideration.

The turtle's tail and head represent the input and the output of the process. Firstly the overall risk of the process for the company and the risk from the cross over points to other processes should be considered.

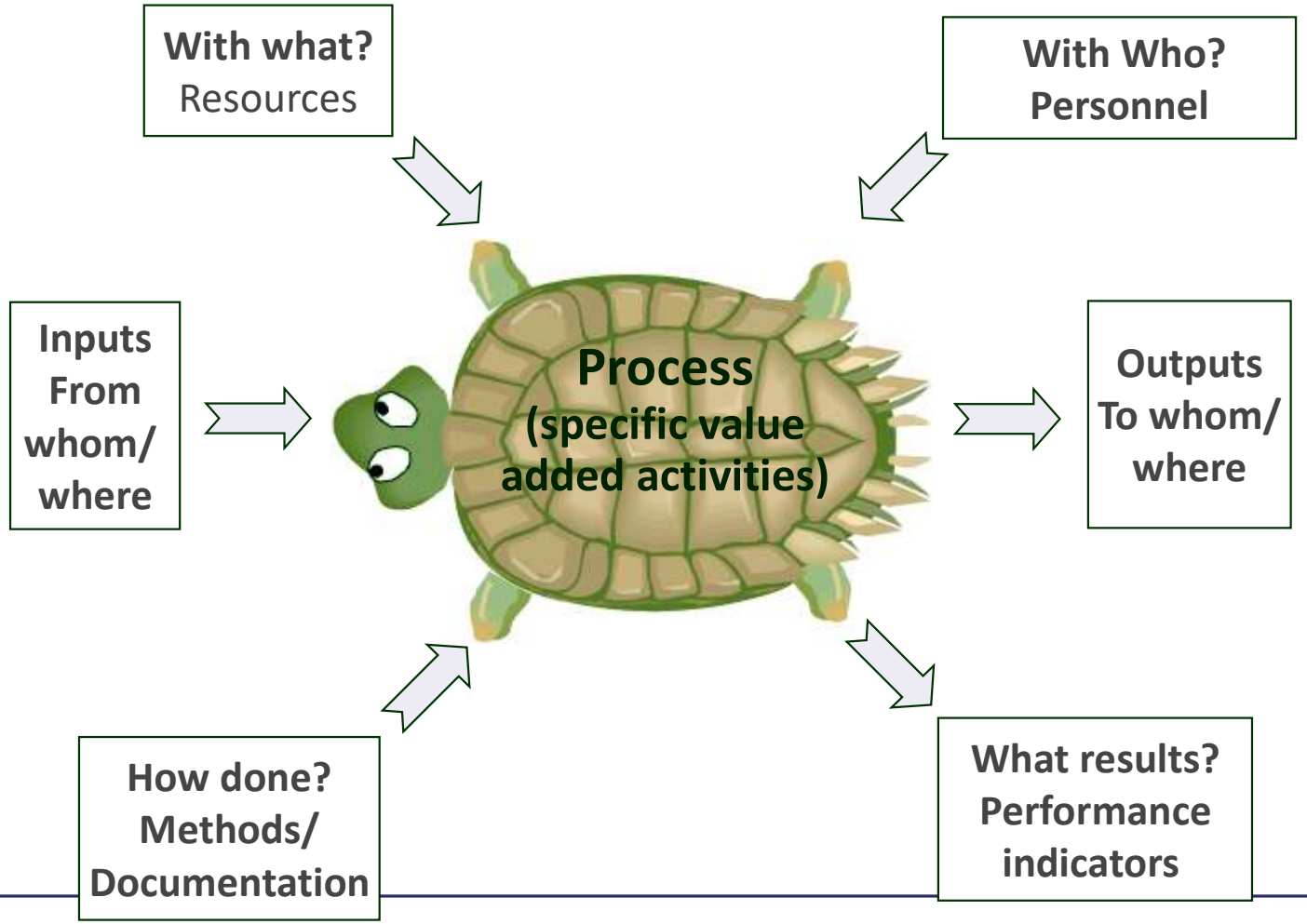
The turtle's four legs represent the pillars of the process:



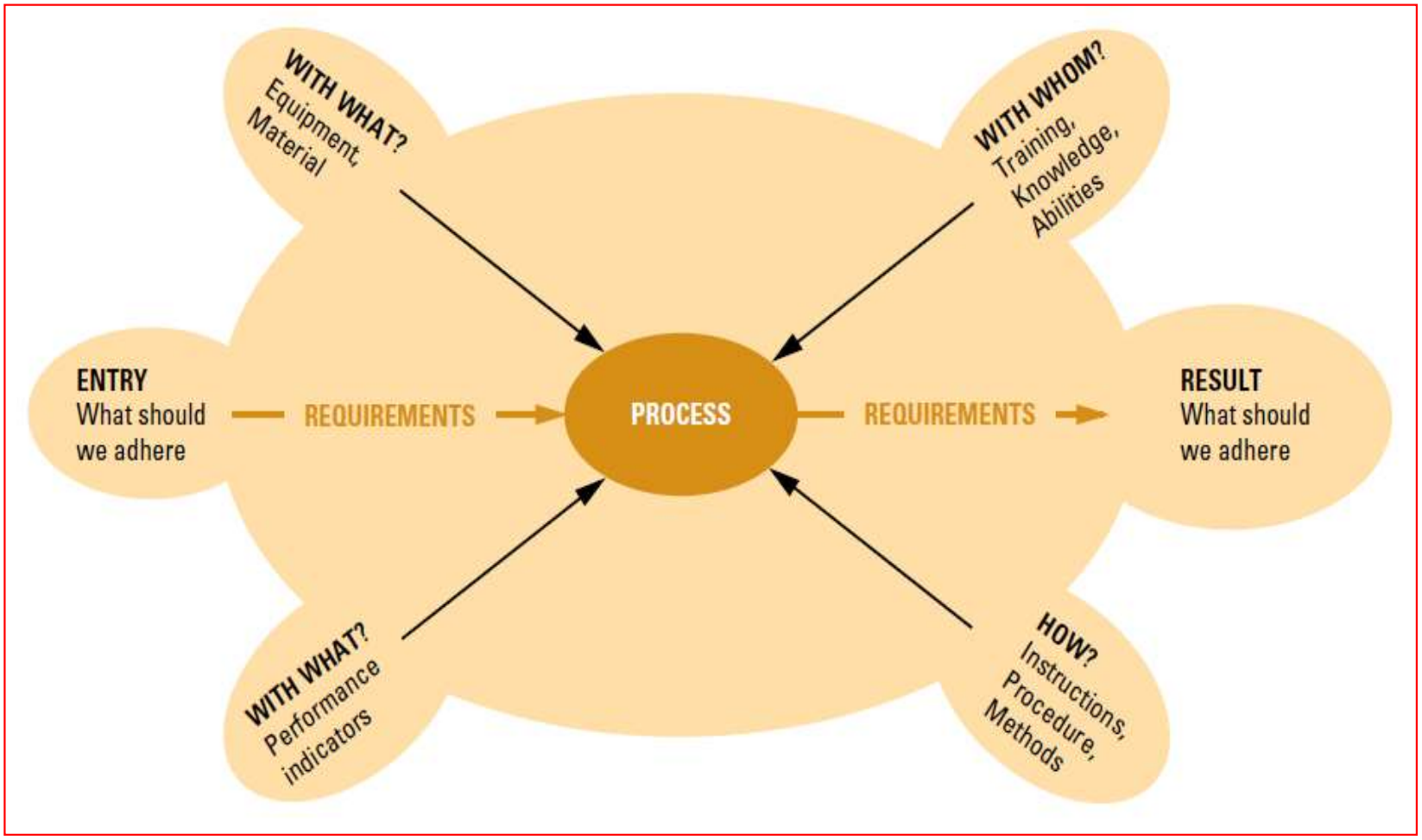
1. The **material resources** (equipment, material etc.)
2. The **staff resources** (qualified personal)
3. Available **Know-how** (procedural descriptions, methods, techniques)
4. **Performance indicators** for the processes under examination.

These individual process components harbor risks, which must be recognized and dealt with accordingly (if necessary, by establishing additional management and/or supporting processes).

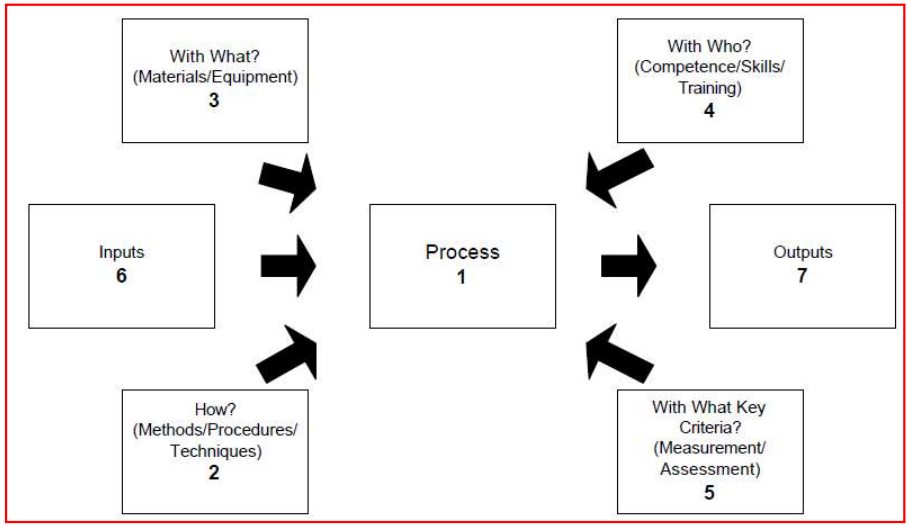
The Turtle Diagram



The Turtle Diagram

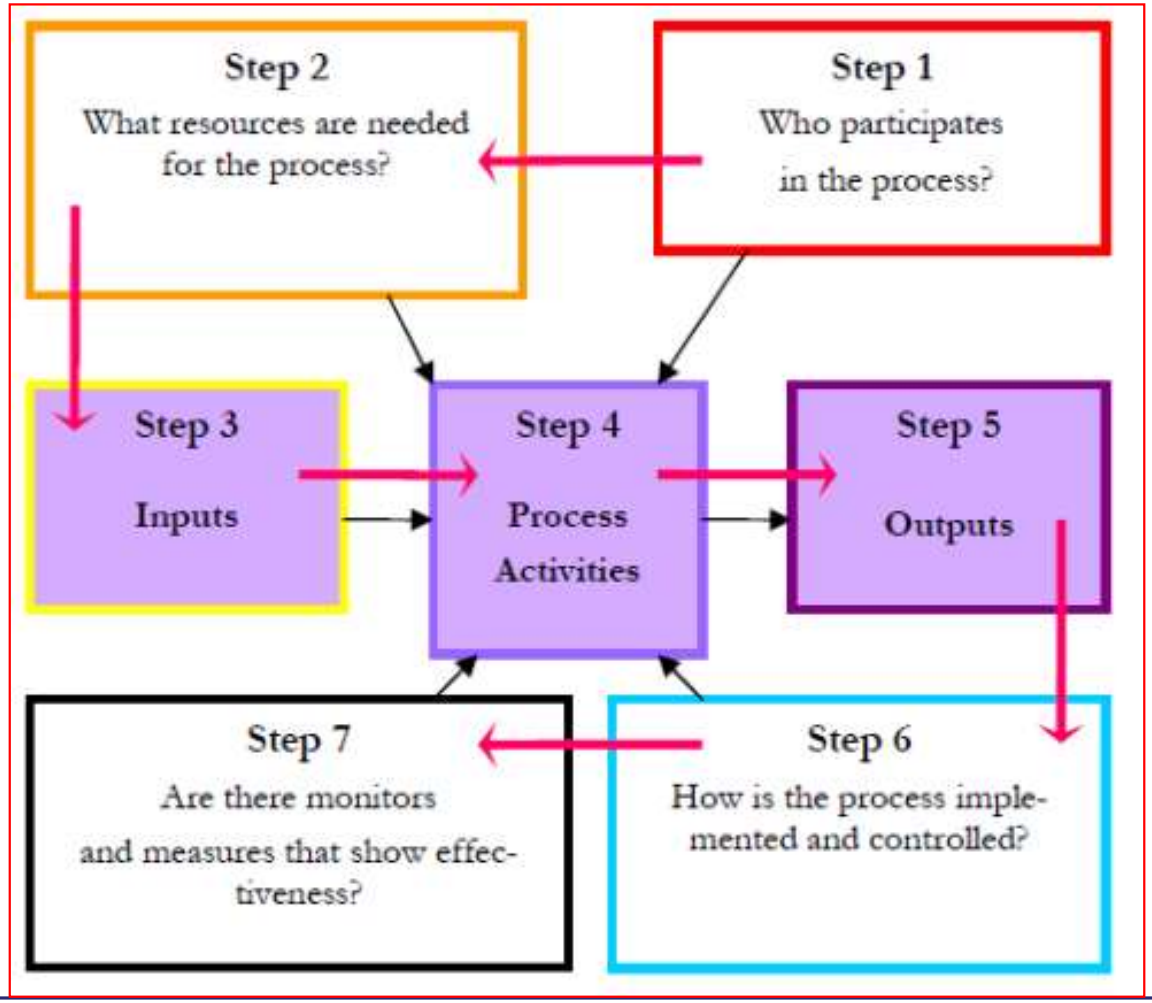


The steps: Method 1

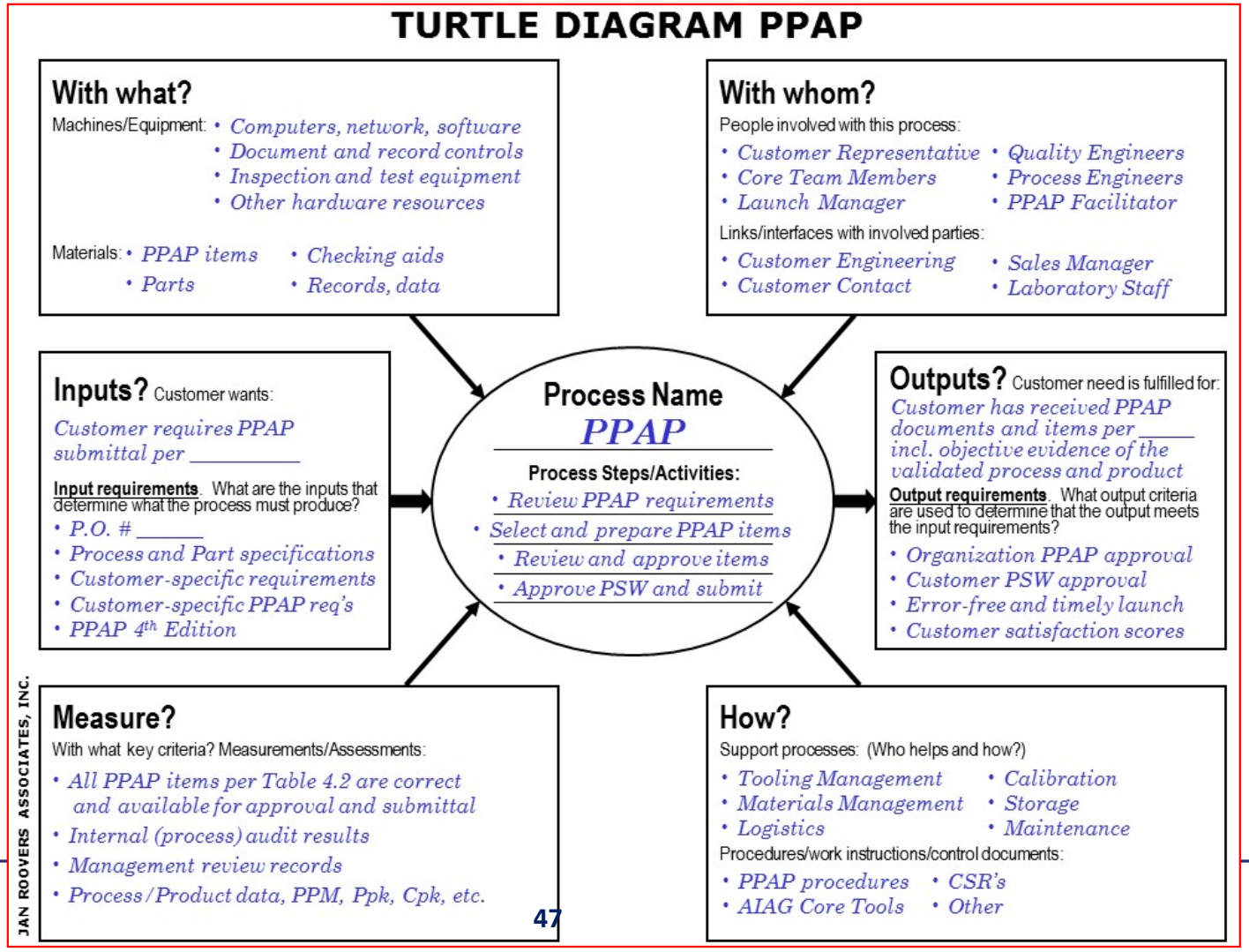


Section	Details
1	Enter Process or Support Process name
2	Enter details of linked process controls, support process, procedures, methods etc.
3	Enter details of the machine, materials (including test equipment), computer systems, software used in the process
4	Enter resource requirements, pay particular attention to required skills and competence criteria, safety equipment etc.
5	Enter the measures of process effectiveness i.e. matrix and target
6	Enter details of the actual input this may be a document, materials, tooling, schedule etc.
7	Enter details of the actual output this may be a product, document, and should be linked to actual measure of effectiveness

The steps: Method 2

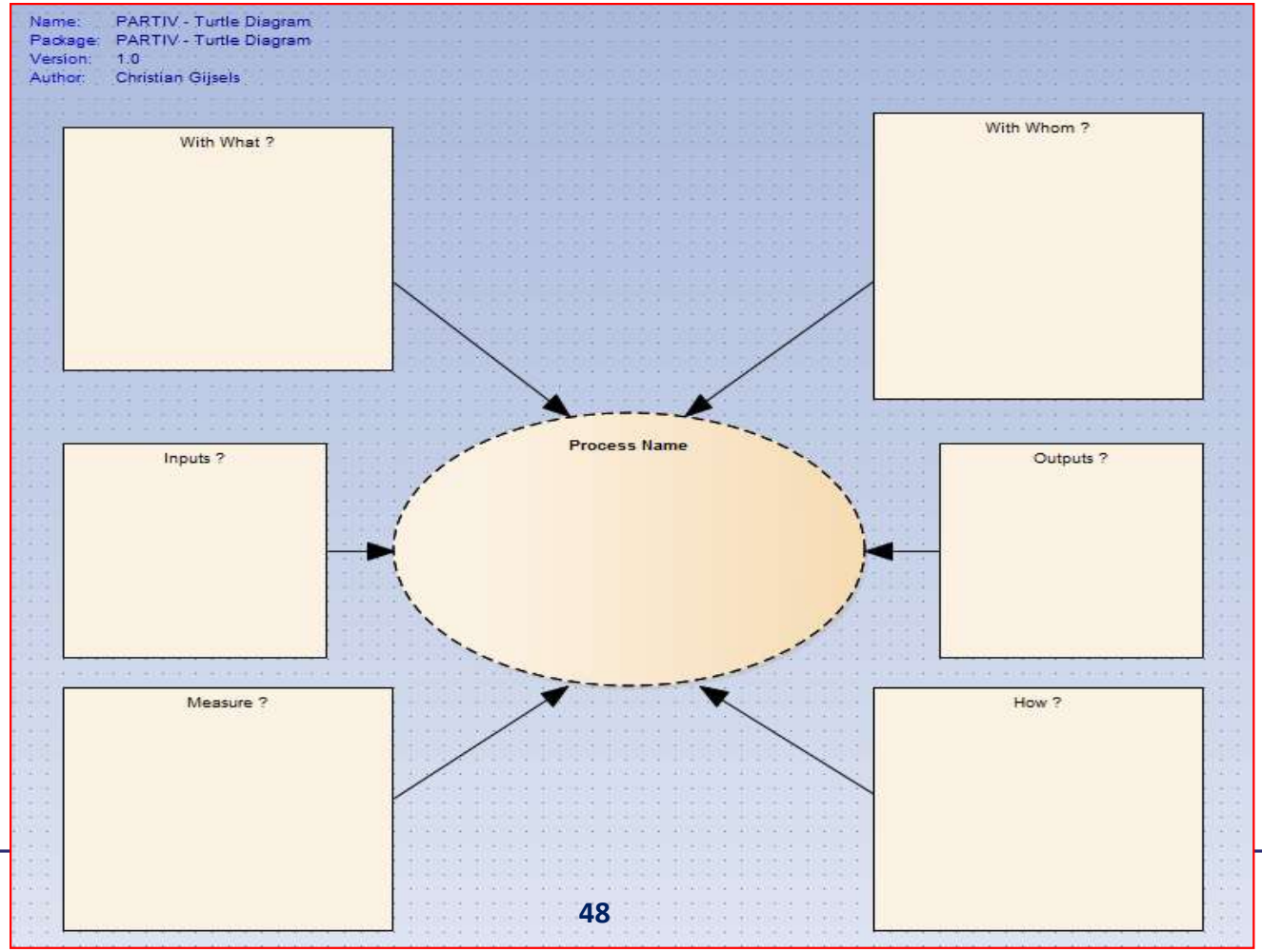


Examples

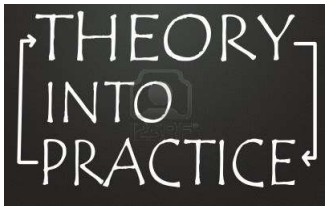

TURTLE DIAGRAM PPAP
With what?
Machines/Equipment:
• Computers, network, software
• Document and record controls
• Inspection and test equipment
• Other hardware resources
Materials:
• PPAP items
• Parts
• Checking aids
• Records, data
With whom?
People involved with this process:
• Customer Representative
• Core Team Members
• Launch Manager
• Quality Engineers
• Process Engineers
• PPAP Facilitator
Links/interfaces with involved parties:
• Customer Engineering
• Customer Contact
• Sales Manager
• Laboratory Staff
Inputs? Customer wants:
Customer requires PPAP submittal per _____
Input requirements. What are the inputs that determine what the process must produce?
• P.O. # _____
• Process and Part specifications
• Customer-specific requirements
• Customer-specific PPAP req's
• PPAP 4th Edition
Process Name
PPAP
Process Steps/Activities:
• Review PPAP requirements
• Select and prepare PPAP items
• Review and approve items
• Approve PSW and submit
Outputs? Customer need is fulfilled for:
Customer has received PPAP documents and items per _____ incl. objective evidence of the validated process and product
Output requirements. What output criteria are used to determine that the output meets the input requirements?
• Organization PPAP approval
• Customer PSW approval
• Error-free and timely launch
• Customer satisfaction scores
Measure?
With what key criteria? Measurements/Assessments:
• All PPAP items per Table 4.2 are correct and available for approval and submittal
• Internal (process) audit results
• Management review records
• Process/Product data, PPM, Ppk, Cpk, etc.
How?
Support processes: (Who helps and how?)
• Tooling Management
• Materials Management
• Logistics
• Calibration
• Storage
• Maintenance
Procedures/work instructions/control documents:
• PPAP procedures
• AIAG Core Tools
• CSR's
• Other
JAN ROOVERS ASSOCIATES, INC.

Turtle diagram – Demo

Demo with Enterprise Architect



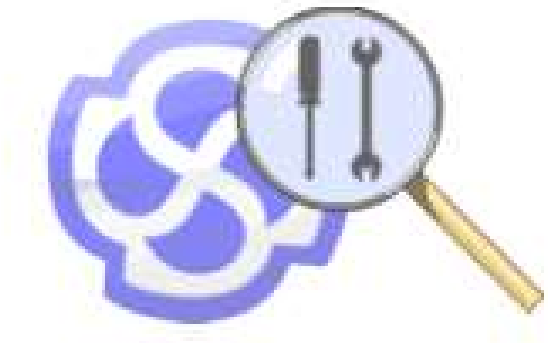
Turtle diagram - Exercise



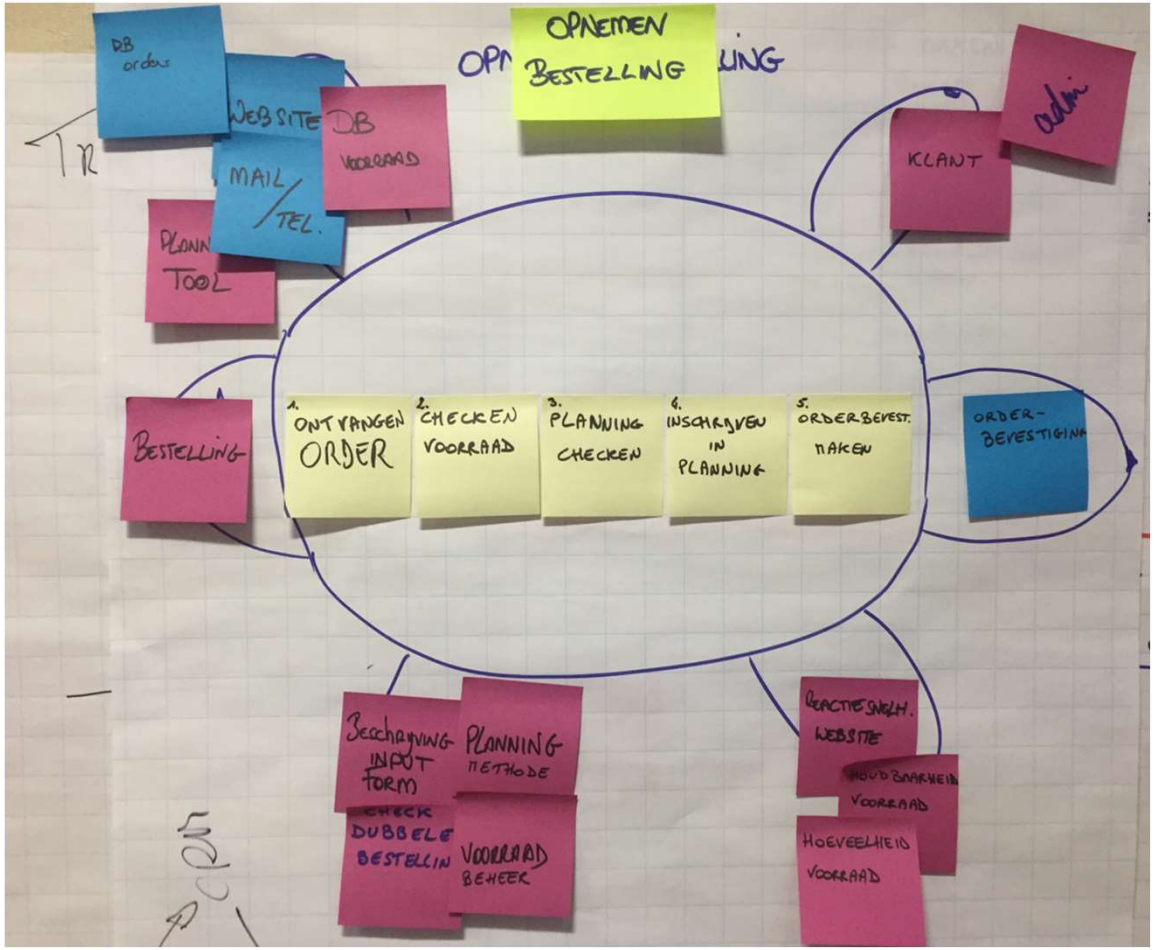
Exercise: Make a Turtle diagram Diagram

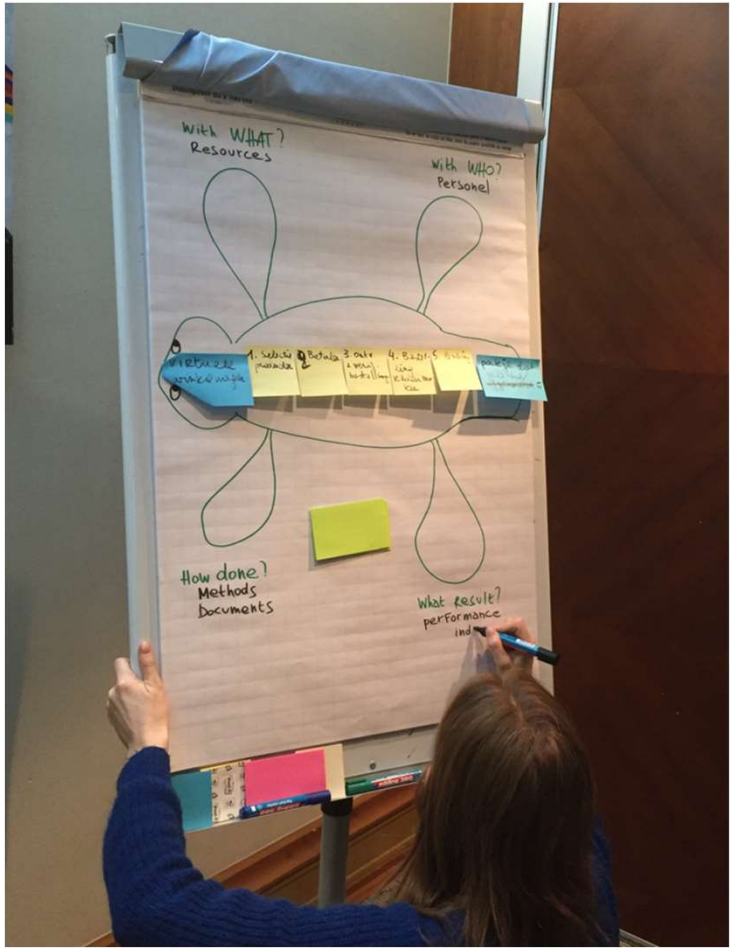
- Model the Turtle Diagram of the Brown Paper Session with your colleague where you both were involved in.

- Present your Turtle Diagram



Turtle diagram - Examples





Part V: RACI Diagram

- Introduction
  - A story
  - What is it
- About
- How
- RACI
- Software
- Cases
- Exercise

RACI Diagram - Introduction

A story

- There were four groups of people named:
  - **EVERYBODY**

  - **SOMEBODY**

  - **ANYBODY**

  - **NOBODY**

RACI Diagram - Introduction

A story

- There was an important job to be done and **EVERYBODY** was asked to do it…
- **EVERYBODY** was sure that **SOMEBODY** would do it…
- **ANYBODY** could of done it, but **NOBODY** did it!
- **SOMEBODY** got angry about that, because it was **EVERYBODY's** job!
- **EVERYBODY** thought that
  ANYBODY could do it, but
  NOBODY realised that
  EVERYBODY wouldn't do it

RACI Diagram - Introduction

A story

- It ended up that **EVERYBODY** blamed **SOMEBODY** when **ANYBODY** could of done what **NOBODY** did …

…. sound familiar?

RACI Diagram - Introduction

What is it

- It is a 6SIGMA business and communication improvement tool.
- A tool to help us plan who does what and when.

RACI Diagram - About

What is the purpose

- To help prevent the major causes of conflict and frustration in most organisations:
  - Poor communication
  - Misunderstanding
  - Lack of information

RACI Diagram - About

How does it help us

- Helps determine:
  - What activities, functions and tasks must be done and who must do them
  - Misunderstanding
  - Lack of information
- It is a charting process to help:
  - Identify, plan, manage and complete jobs
- It is highly participative

RACI Diagram - About

Why do we need it

- To some up, some common pitfalls:
  - Nobody does the job.
  - Too many try to do it.
  - There is confusion over roles.
  - Someone ends up doing everything.

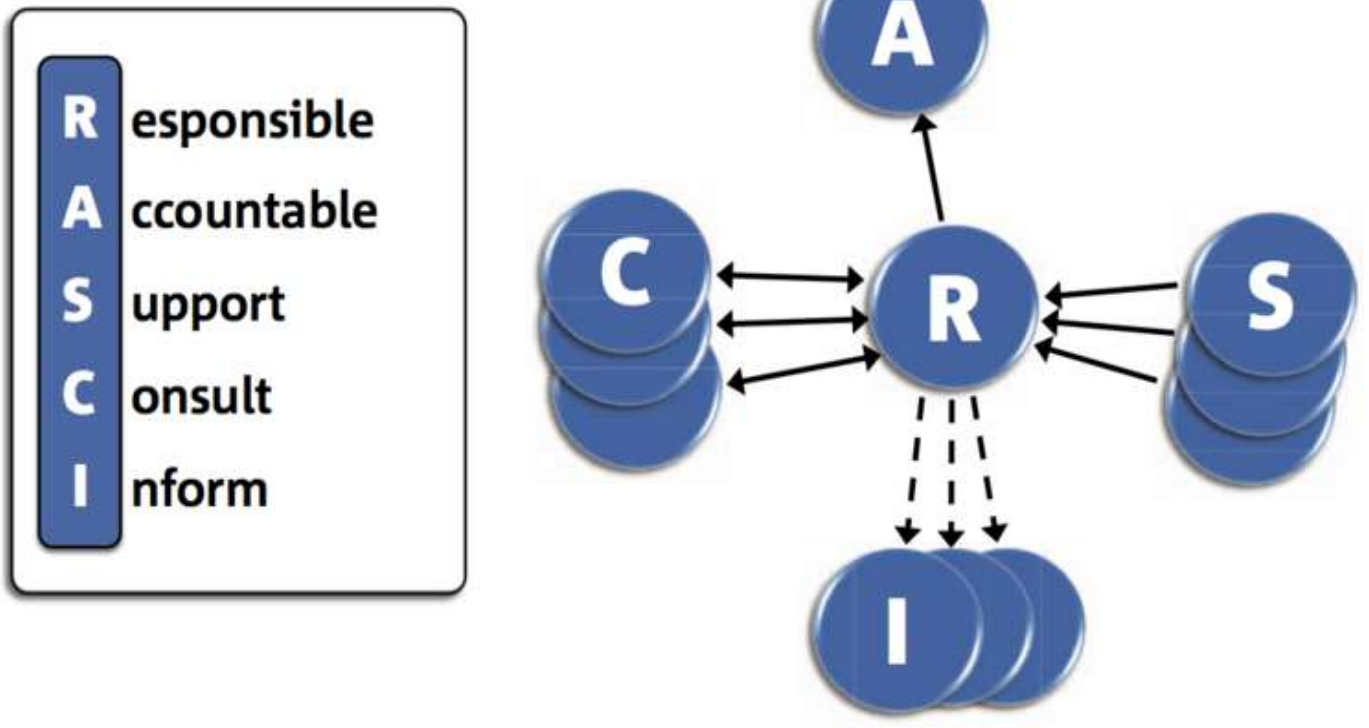
RACI Diagram - About

Why do we need it

- Needs to be done so that:
  - Agreement is reached on what jobs need to be done.
  - Accountabilities are defined and agreed.
  - Communication is improved.
  - Duplication of effort is avoided.
  - Jobs are done properly, on time.
  - Finger pointing and stress are avoided.

RACI Diagram - How

How can RA(S)CI do all this ?



RACI Diagram – RACI

RESPONSIBLE

- The individual(s) who actually do the job.
- The degree of responsibility is defined by the accountable person.
- Responsibilities can be shared.

RACI Diagram – RACI

ACCOUNTABLE

- The individual who is ultimately accountable their neck is on the line if the job isn't done!
- They have the power of veto.
- Only one "A" can be assigned to an activity or decision.
- Can delegate the "R".

RACI Diagram – RACI

CONSULTED

- The individual(s) who need to be consulted prior to a final decision or action being taken.
- This is TWO-WAY communication.
- Consulted may not have a direct part in the task but is impacted by its completion.
- Their input may be necessary.

RACI Diagram – RACI

INFORMED

- The individual(s) who need to be informed after a decision or action is taken.
- This is ONE-WAY communication.
- Input from the informed party is not necessary – buy they will need to know.

RACI Diagram – RACI

The benefits

- Improved:
  - Teamwork.
  - Cooperation.
  - Communication.
  - Productivity.

- Less:
  - Frustration.
  - Duplication of effort.

RACI Diagram – RACI

The benefits

- Improved:
  - Teamwork.
  - Cooperation.
  - Communication.
  - Productivity.

- Less:
  - Frustration.
  - Duplication of effort.

RACI Diagram – Cases

Case

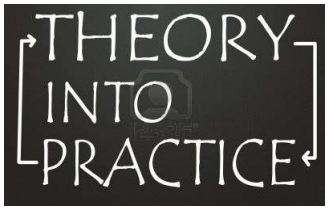
- These RACI is used to indicate who in the team has which specific function for each deliverable.

Function	Project Sponsor	Project Manager	Developer	Analyst
Project Initiation	C	A R		
Establish Project Plan	I	A R	C	C
Collate User Requirement	I	A	I	R
Define Technical Requirements	I	A	I	R
Develop Software Tools	I	A	R	C
Test Software	I	A	C	R
Install Software	C	A	C	R

- R - Responsible for doing the work
- A - Accountable for the work being done
- C - Must be consulted for input
- I - Must be kept informed of progress and results

- This example chart shows the work to be done in the left column as activities with the assigned resources along the top. The RACI is useful when the team consists of internal and external resources to ensure clear divisions of roles and expectations

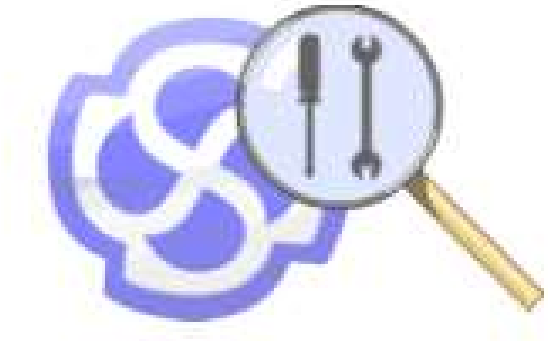
RACI Diagram - Exercise



Exercise: Make a RACI Diagram

- Model a RACI Diagram

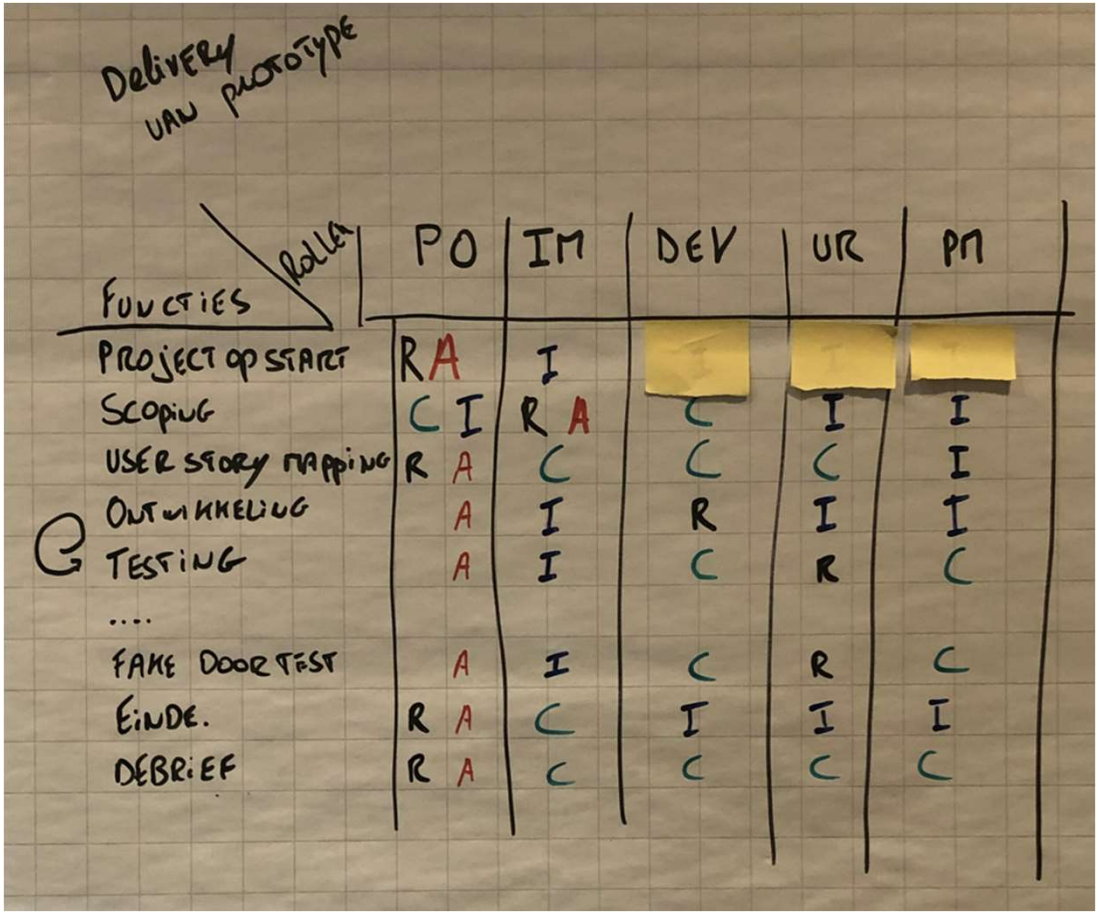
- Present your RACI Diagram

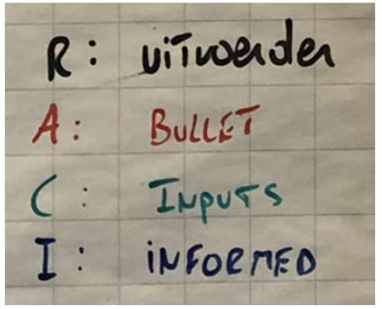


RACI Diagram – Examples

Verantwoordelijkheidsschema bouwproject (Raci model) datum:					**functies en personen**													
project:	*projectnaam*			**R = voert werk uit**														
opdrachtgever:	*naam*			**A = eindverantwoordelijk**														
projectleider:	*naam*			**C = betrokken**														
projectstart	*datum*			**i = geinformeerd**														
projecteind	*datum*																	
projectfasering		**taakgebieden**	**no**	**activiteiten**	Opdrachtgever	Projectleider	Projectteamlid 2	Projectteamlid 2	Architect	Adviseurs	Aannemers	Interne organisatie	Eindgebruikr 1	Eindgebruiker 2	Belanghebbenden			
INITIATIEFFASE		**Projectdefinitie**	1	initiatief	R													
		(initiatiefopdracht)	2	haalbaarheidsonderzoek	R													
			3	toets bstemmingsplan	A	R												
			4	vaststellen budget	A	R												
			5	programma van eisen	A	R												
			6	projectdefinitie (opdracht)	R													
DEFINITIEFASE		**Projecplan**	5	projectplan opstellen		R	C	C										
		(uitvoeringsopdracht)	6	tijdsplanning		R	C	C										
			7	kostenraming		R	C	C										
			8	organisatie / omgeving		R	C	C										
			9	kwaliteitsplan		R	C	C										
			10	communicatieplan		R	C	C										
			11	risicoanalyse		R	C	C										
			12	overleg opdrachtgever	R	R	C	C										
			13	overleg eindgebruikers	i	R	C	C					C	C				
			14	overleg projectteam	i	R	R	R										
			15	communicatie en voortgang	i	A	R	C				i	i	i				
			16	informatiebijeenkomsten	i	R	C	C				i	i	i	i			
			17	exploitatie en VvE beheer														
			18	communicatie en voortgang														
			19															
			20															
ONTWERPFASE		**Ontwerp en Bestek**	21	aanbesteding architect	i	A	C	C	C	R		C	i	i	i			

RACI Diagram – Examples





Part VI: Data flow modelling in DFD / Data Flow Programs

- Positioning
  - Structured Systems Analysis and Design Methodology (SSADM)
- What is a DFD
- Principles
- The Levels
  - The context diagram
  - Level 0 (The top) Diagram
  - Level 1 Diagram
- DFD Notations
- DFD Rules
- DFD Styles
- Software
- Case studies
  - Real case examples

- Demo
- Exercise

Positioning

Structured Systems Analysis and Design Methodology (SSADM)

is a systems approach to the analysis and design of information systems. SSADM was produced for the CCTA, a UK government office concerned with the use of technology in government, from 1980 onwards.

The SSADM method involves the application of a sequence of analysis, documentation and design tasks concerned with:

- Analysis of the current system. Also known as: feasibility stage. Analyze the current situation at a high level. A **DFD** (**Data Flow Diagram**) is used to describe how the current system works and to visualize known problems.
- Outline business specification
- Detailed business specification
- Logical data design
- Logical process design
- Physical design

What is a DFD

What is a Data Flow Diagram

A **Data Flow Diagram** or **DFD** is a graphical representation of **the "flow" of data through an information system**. A data flow diagram can also be used for the visualization of data processing (structured design).

It is common practise for a designer to draw a **context-level DFD** first which shows the interaction between the system and outside entities. This context-level DFD is then "exploded" to show more detail of the system being modelled.

Data flow diagrams do not show decisions or timing of events. Their function is to illustrate data sources, destinations, flows, stores, and transformations. The capabilities of data flow diagramming align directly with general definitions of systems.

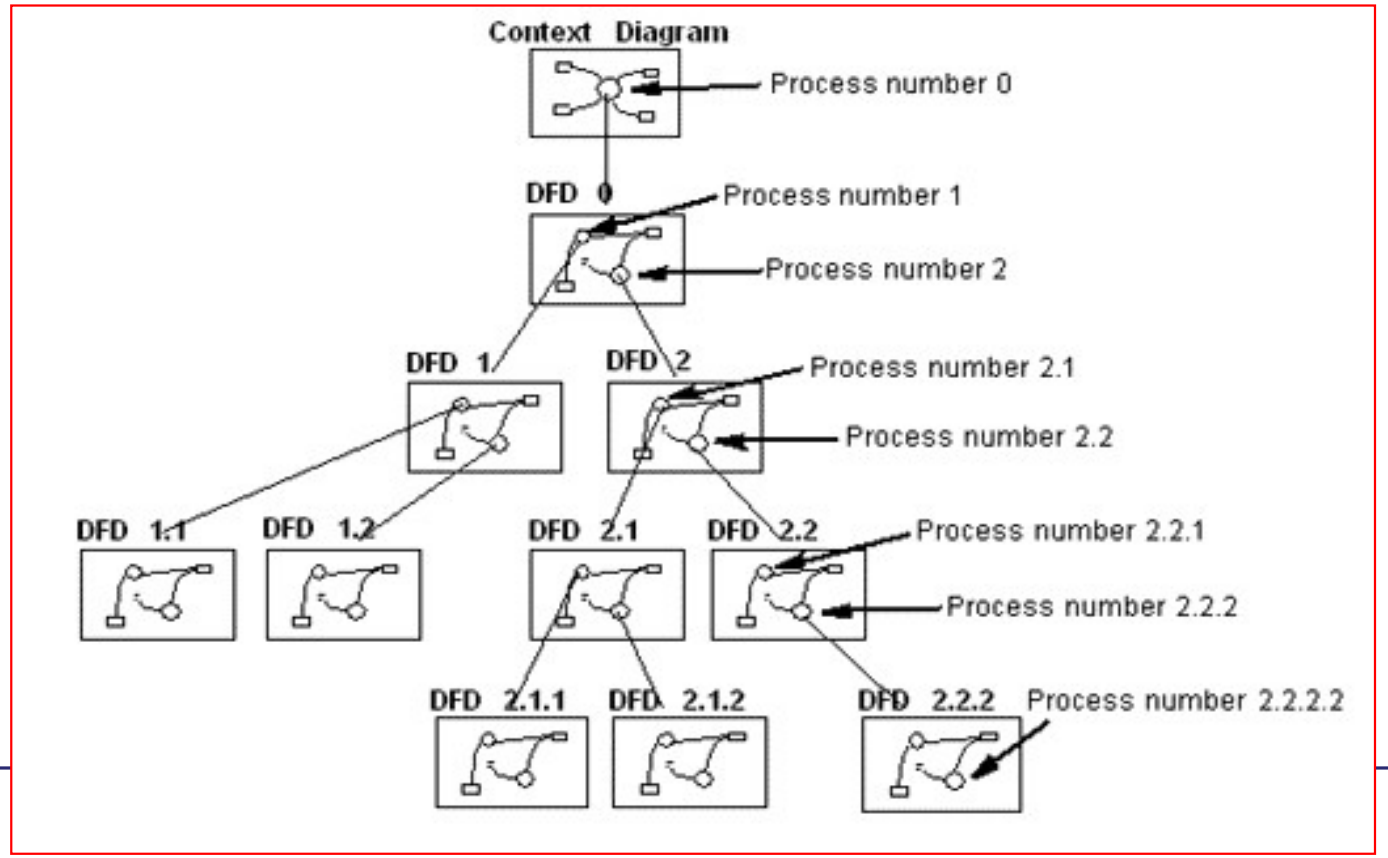
Data flow diagrams are an implementation of a method for representing systems concepts including boundaries, input/outputs, processes/sub processes, etc.

Sometimes a DFD diagram and an BPMN diagram are both used in a business analysis.

Principles

Principles of a Data Flow Diagram

The general principle in Data Flow Diagramming is that a system can be decomposed into **subsystems**, and **subsystems** can be decomposed into **lower level subsystems**, and so on.

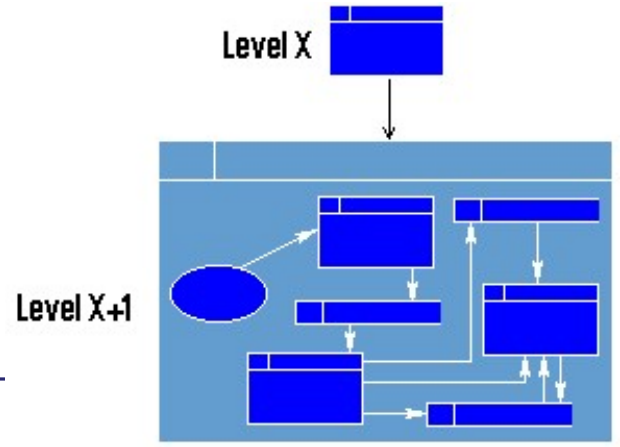


Principles

Principles of a Data Flow Diagram

- Each subsystem represents a process or activity in which data is processed. At the lowest level, processes can no longer be decomposed.
- Each 'process' (and from now on, by 'process' we mean subsystem and activity) in a DFD has the characteristics of a system.
- Just as a system must have input and output (if it is not dead), so a process must have input and output.
- Data enters the system from the environment; data flows between processes within the system; and data is produced as output from the system.

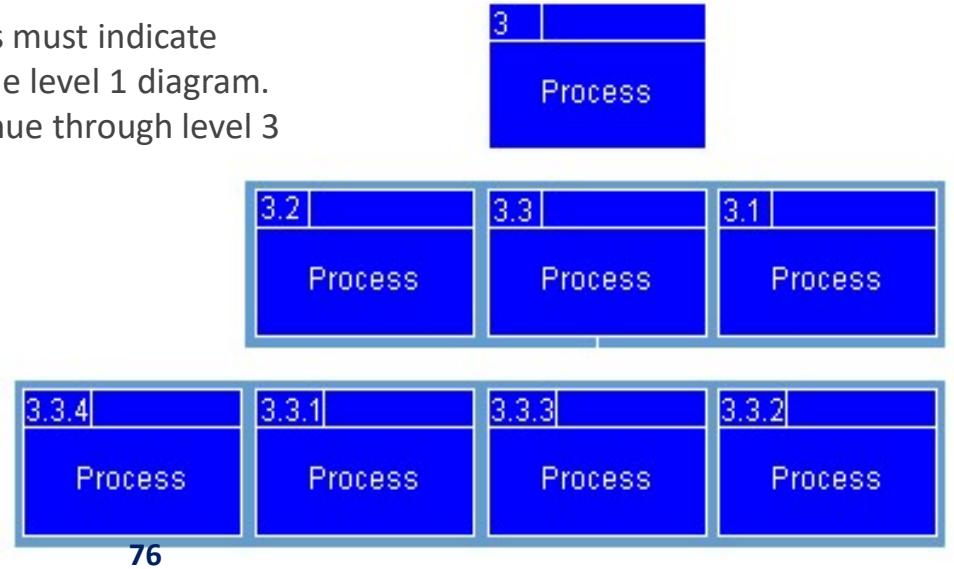
- Top Down expansion
  - Each process within a given business process diagram may be the subject of further analysis. This involves identifying
    the lower level processes that together constitute the process as it was originally identified. This procedure is known as top-down expansion or levelling.
  - As a business process diagram is decomposed, each process box becomes a boundary for the next, lower level, diagram.



Principles

Principles of a Data Flow Diagram

- Numbering Rules
  - The process boxes on the level 1 diagram should be numbered arbitrarily, so that no priority is implied. Even where data from one process flows directly into another process, this does not necessarily mean that the first one has to finish before the second one can begin.
  - Therefore the processes on a level 1 diagram could be re-numbered without affecting the meaning of the diagram. This is true within any business process diagram - as these diagrams do not imply time, sequence or repetition.
  - However, as the analysis continues beyond level 1 it is important that a strict numbering convention is followed. The processes on level 2 diagrams must indicate their parent process within the level 1 diagram. This convention should continue through level 3 diagrams, and beyond, should that level of analysis ever be required.

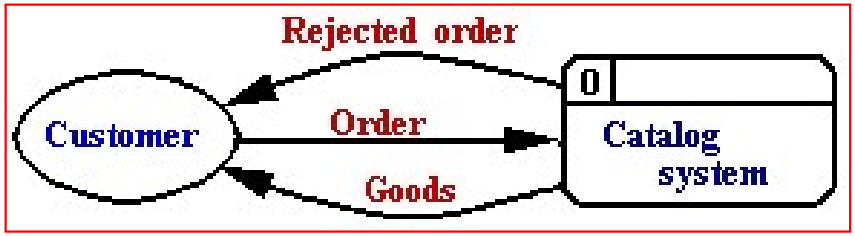


The levels

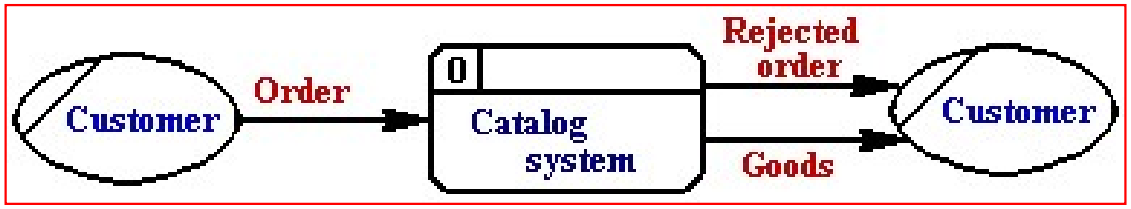
The Context Diagram

The 'Context Diagram' is an overall, simplified, view of the target system, which contains only one process box and the primary inputs and outputs.

Example:



Another representation can be:

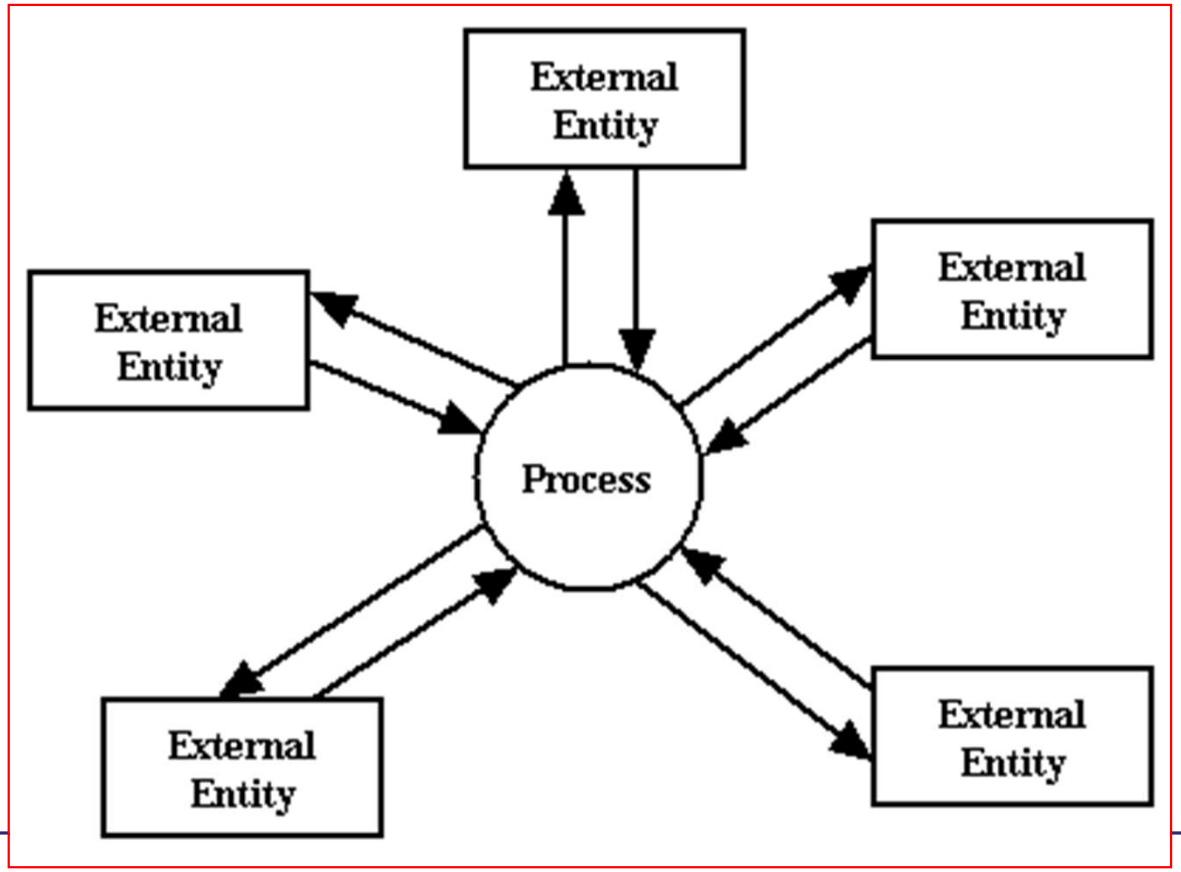


Both the above diagrams say the same thing. The second makes use of the possibility in SSADM (Structured Systems Analysis and Design Methodology) of including duplicate objects. (In the second diagram the duplication of the Customer object is shown by the line at the left hand side. Drawing the diagram in this way emphasizes the Input-Output properties of a system.

The levels

The Context Diagram

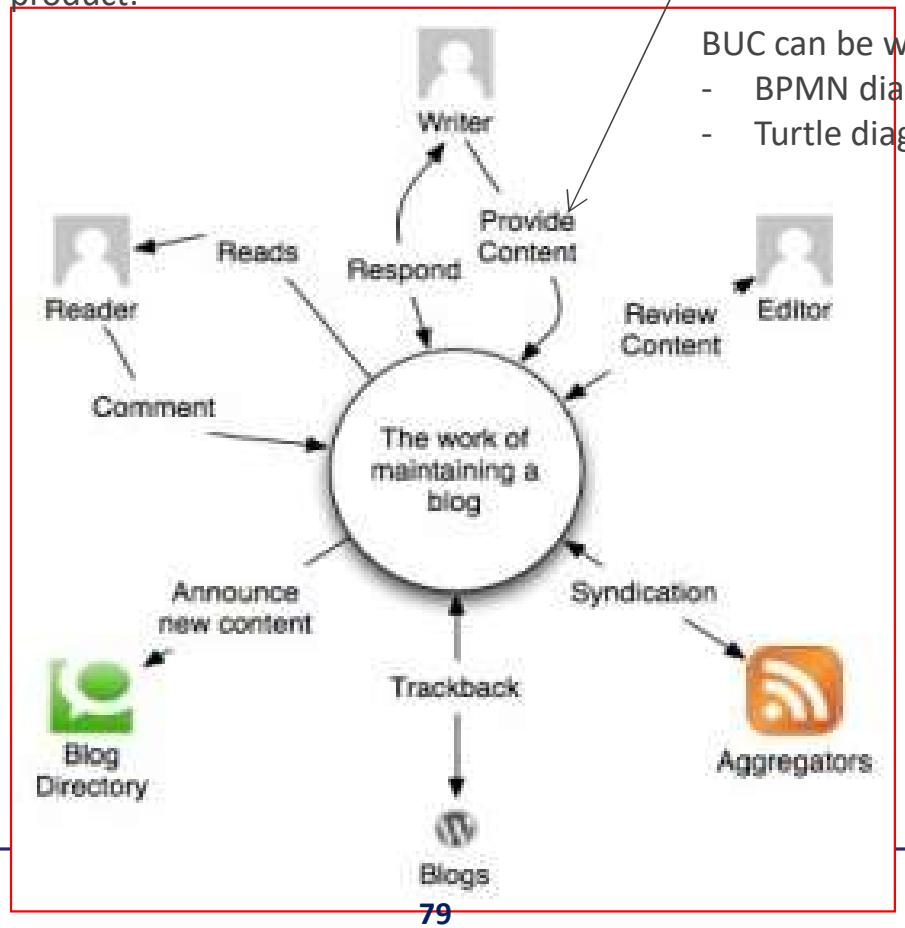
The 'Context Diagram ' general model with multiple External entities:



An example

The Context Diagram

The context of your product:



Provide Content is a Business Event / Business Use Case (BUC)

BUC can be written in details in:

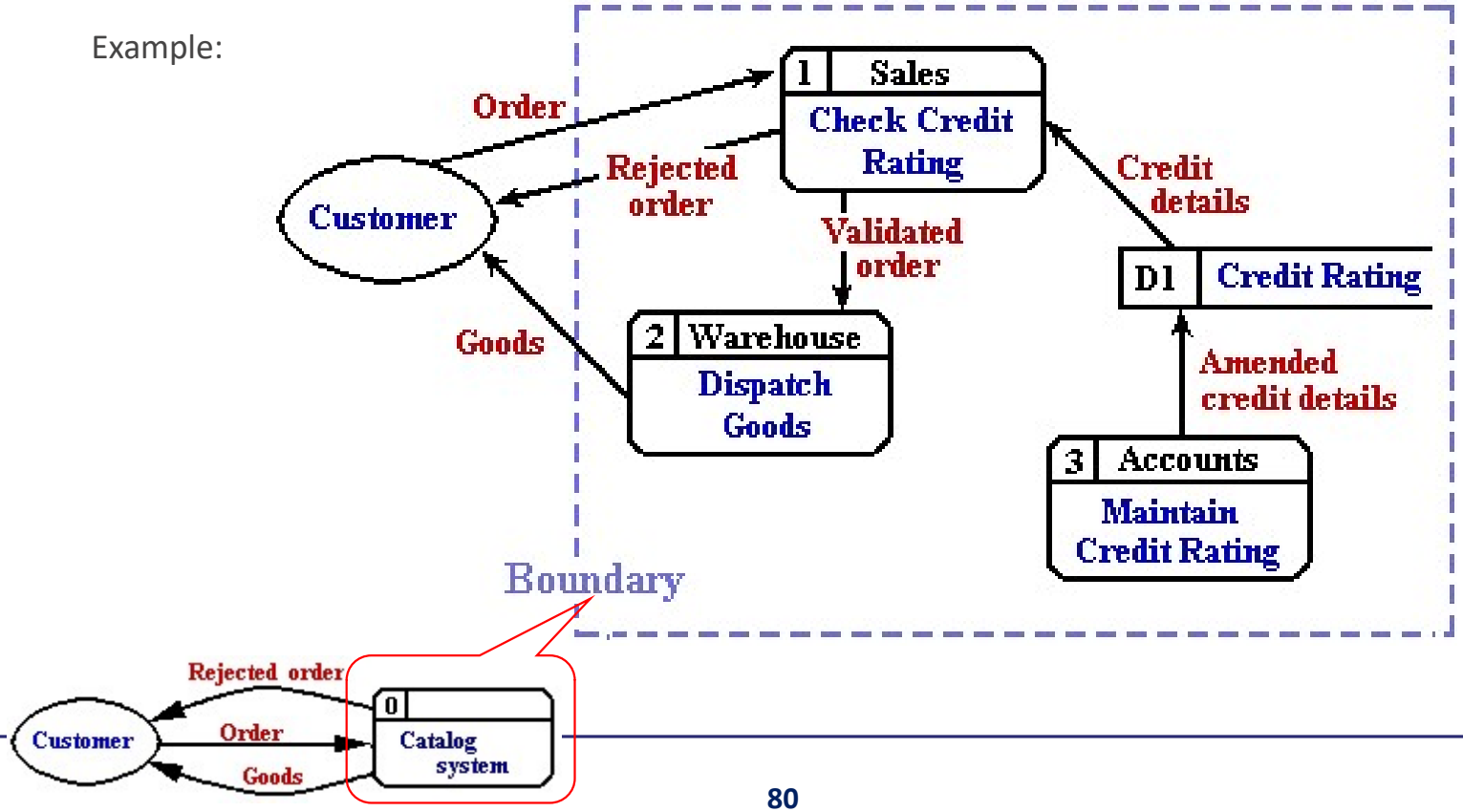
- BPMN diagrams
- Turtle diagrams (less detail)

The levels

The Level 0 (The top) Diagram

The Top or Level 0 diagram, describes the whole of the target system. It 'bounds' the system under consideration.

Example:

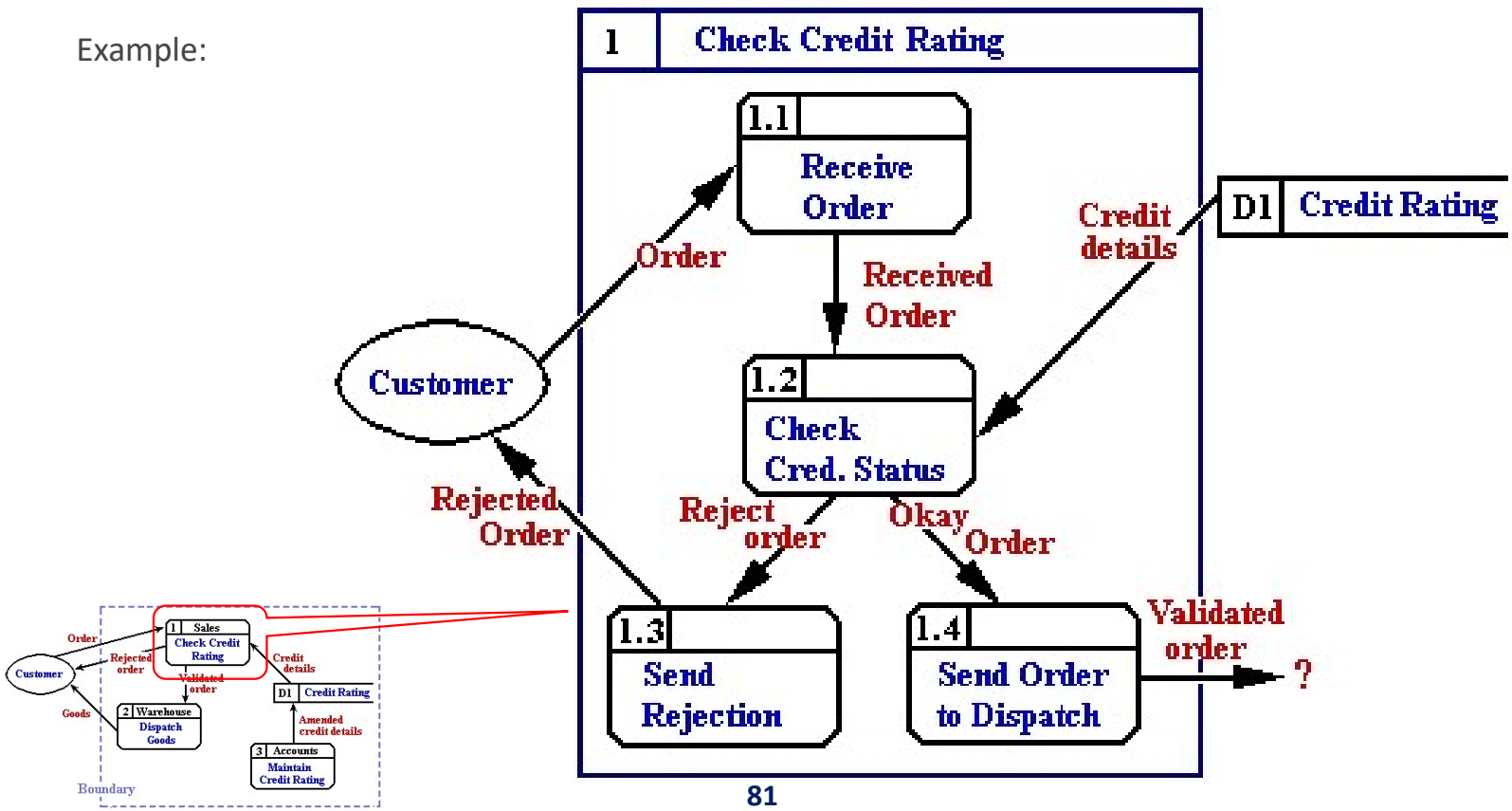


The levels

Level 1 Diagram

Each Process box in the Top Level diagram will itself be made up of a number of processes, and will need to be decomposed as a second level diagram.

Example:



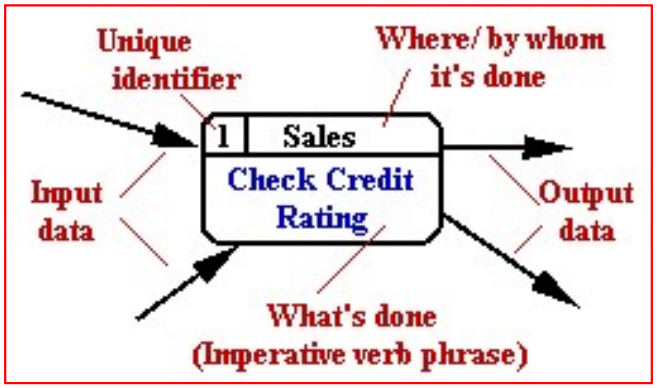
DFD Notations

The Process Symbol

- **Processes** transform or manipulate data. Each box has a unique number as identifier (top left) and a unique name (an imperative - e.g. 'do this' - statement in the main box area) The top line is used for the location of, or the people responsible for, the process.

- **Processes** are 'black boxes' - we don't know what is in them until they are decomposed.

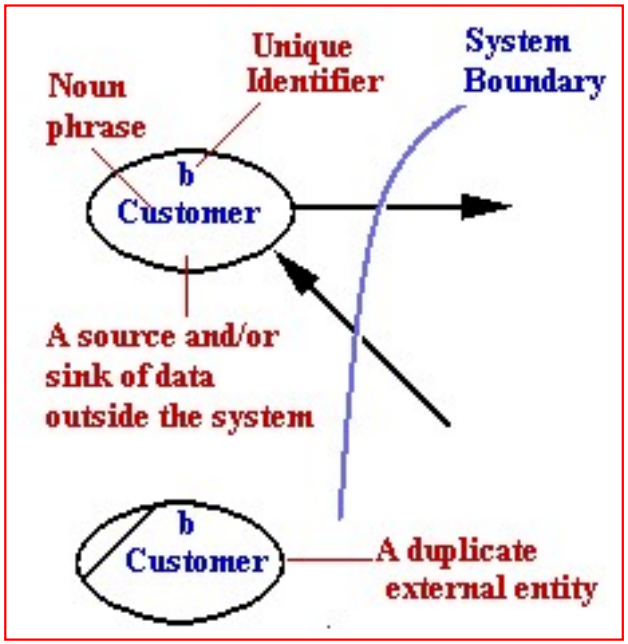
- **Processes** transform or manipulate input data to produce output data. Except in rare cases, you can't have one without the other.



DFD Notations

The External Entities Symbol

- **External Entities**, also known as 'External sources/recipients, are things (e.g.: **people, machines, organisations** etc.) which contribute data or information to the system or which receive data/information from it.
  The name given to an external entity represents a Type not a specific instance of the type. When modelling complex systems, each external entity in a DFD will be given a unique identifier. It is common practice to have duplicates of external entities in order to avoid crossing lines, or just to make a diagram more readable.



DFD Notations

The Resource Flow Symbol

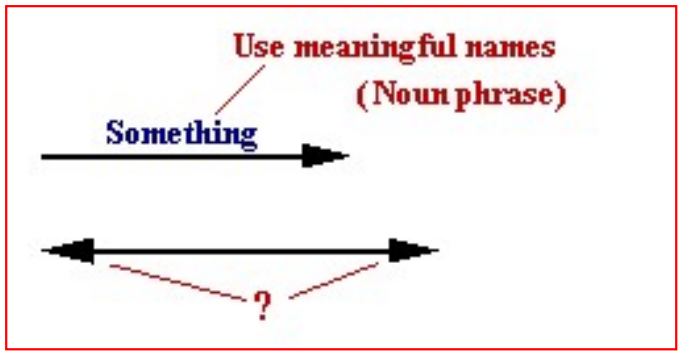
- A **Resource flow** shows the flow of any **physical material** from its source to its destination. For this reason they are sometimes referred to as physical flows.

  The physical material in question should be given a meaningful name. Resource flows are usually restricted to early, high-level diagrams and are used when a description of the physical flow of materials is considered to be important to help the analysis.

DFD Notations

The Data Flow Symbol

- **Data Flows** depict **data/information flowing** to or from a process. The arrows must either start and/or end at a process box. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly. Arrows must be named. Double ended arrows may be used with care.



DFD Notations

The Data Store Symbol

- **Data Stores** are some location where data is held temporarily or permanently.

- In physical DFDs there can be 4 types:

  - **D** = computerised **D**ata
    M = **M**anual, e.g. filing cabinet.
  - **T** = **T**ransient data file, e.g. temporary program file
    T(M) = **T**ransient **M**anual, e.g. in-tray, mail box.

DFD Rules

Data Flow Diagrams rules

- Entities are either 'sources of' or 'sinks' for data input and outputs - i.e. they are the originators or terminators for data flows.
- Data flows from Entities must flow into Processes
- Data flows to Entities must come from Processes
- Processes and Data Stores must have both inputs and outputs
  (What goes in must come out!)
- Inputs to Data Stores only come from Processes.
- Outputs from Data Stores only go to Processes.

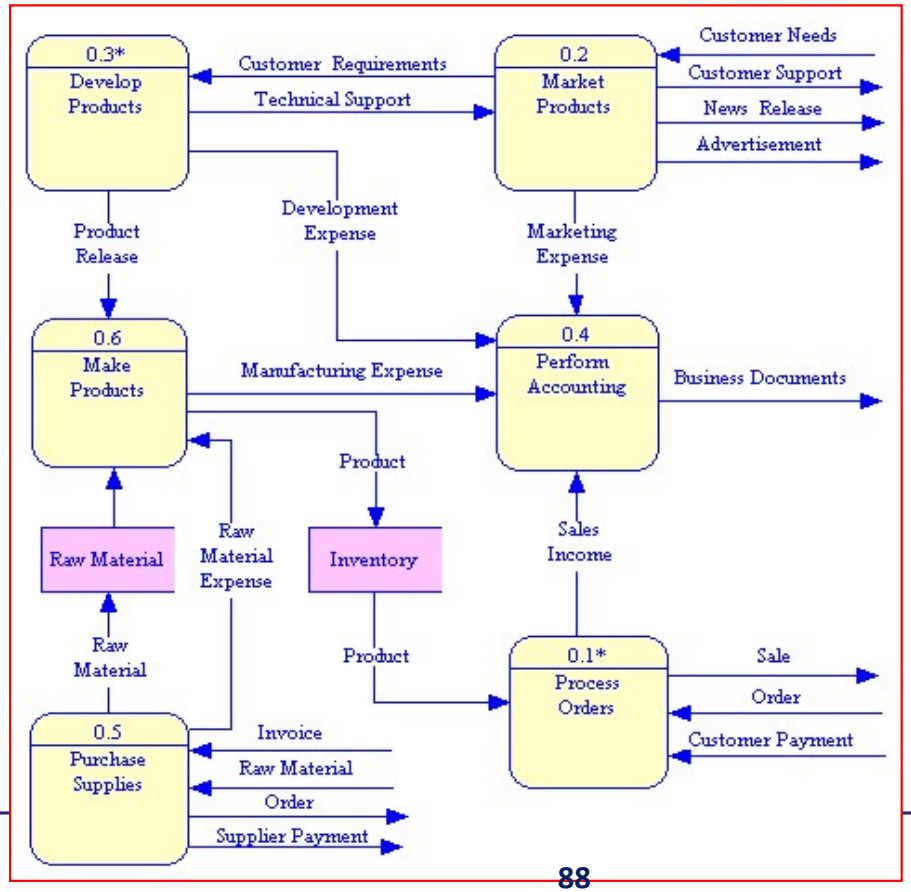
Data Flow Diagrams - Relationship Grid

	External Entity	Process	Data Store
External Entity	✓	✓	✗
Process		✓	✓
Data Store			✓

DFD Styles

Gane and Sarson Diagrams

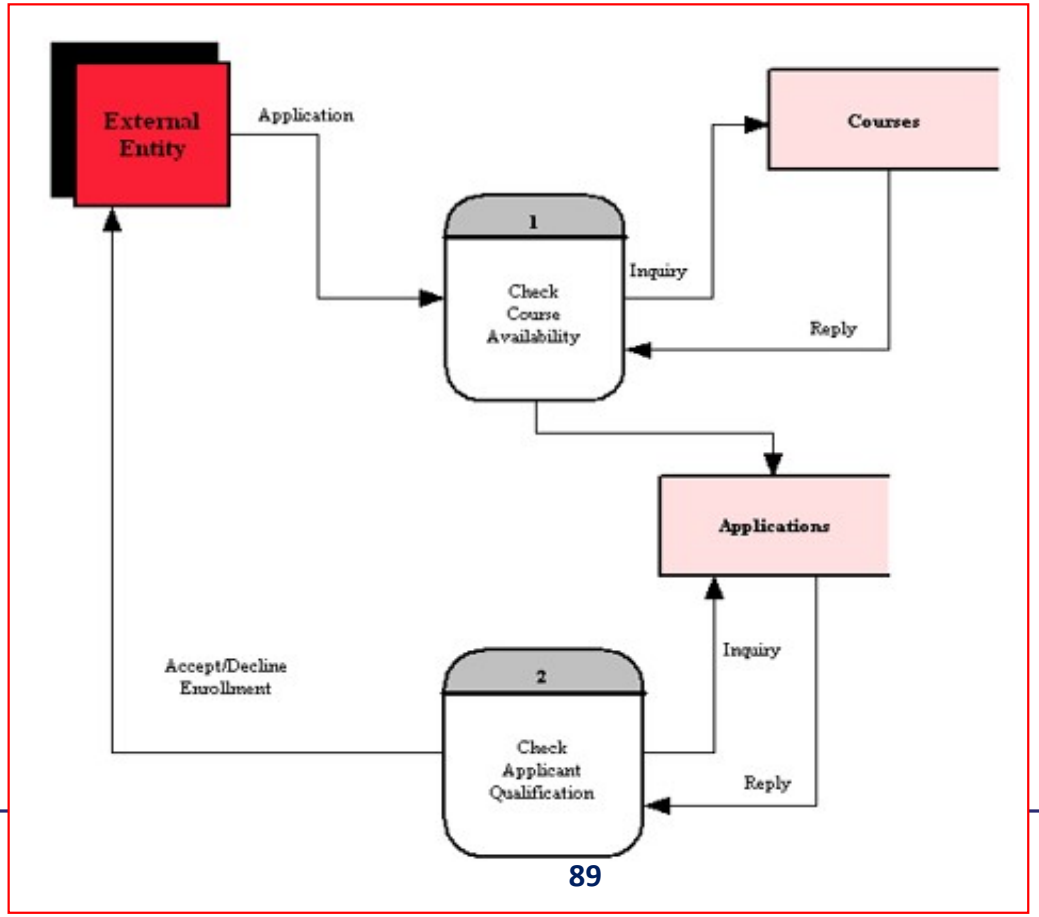
Example: Order to production



DFD Styles

Gane and Sarson Diagrams

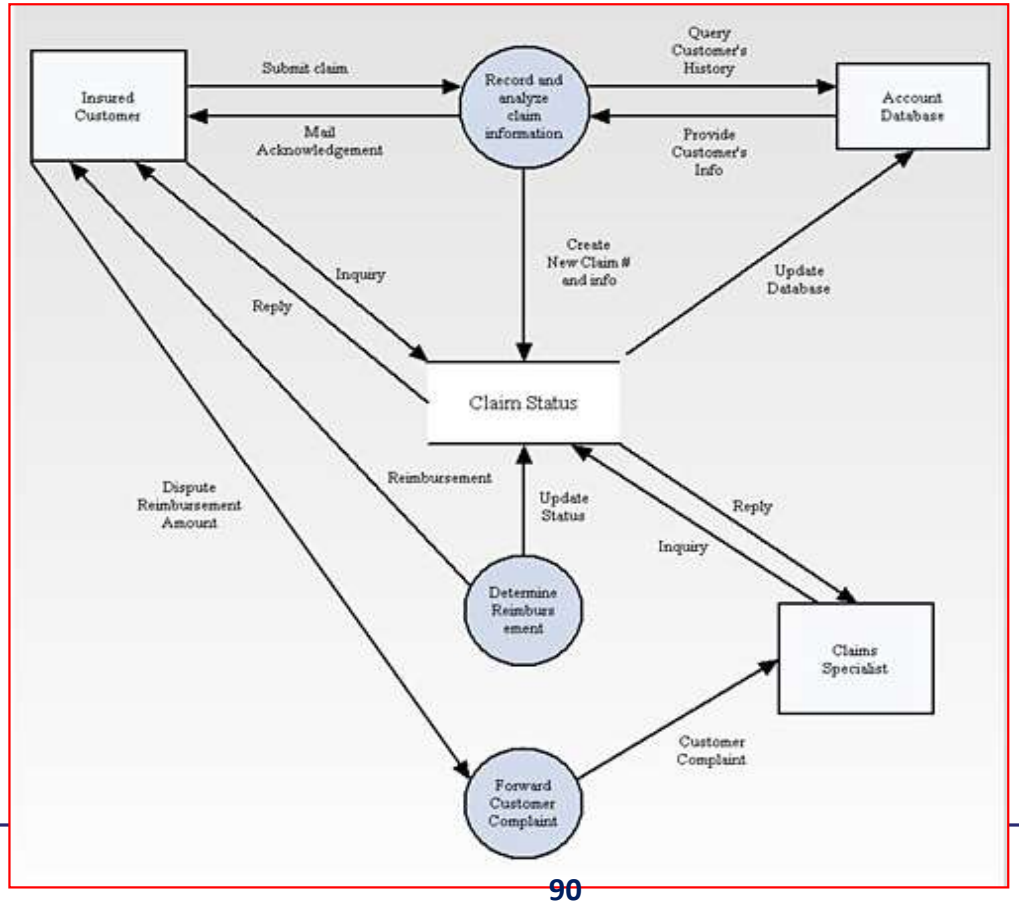
Example: Automated course registration



DFD Styles

Coad and Yourdon Diagrams (the Yourdon symbols)

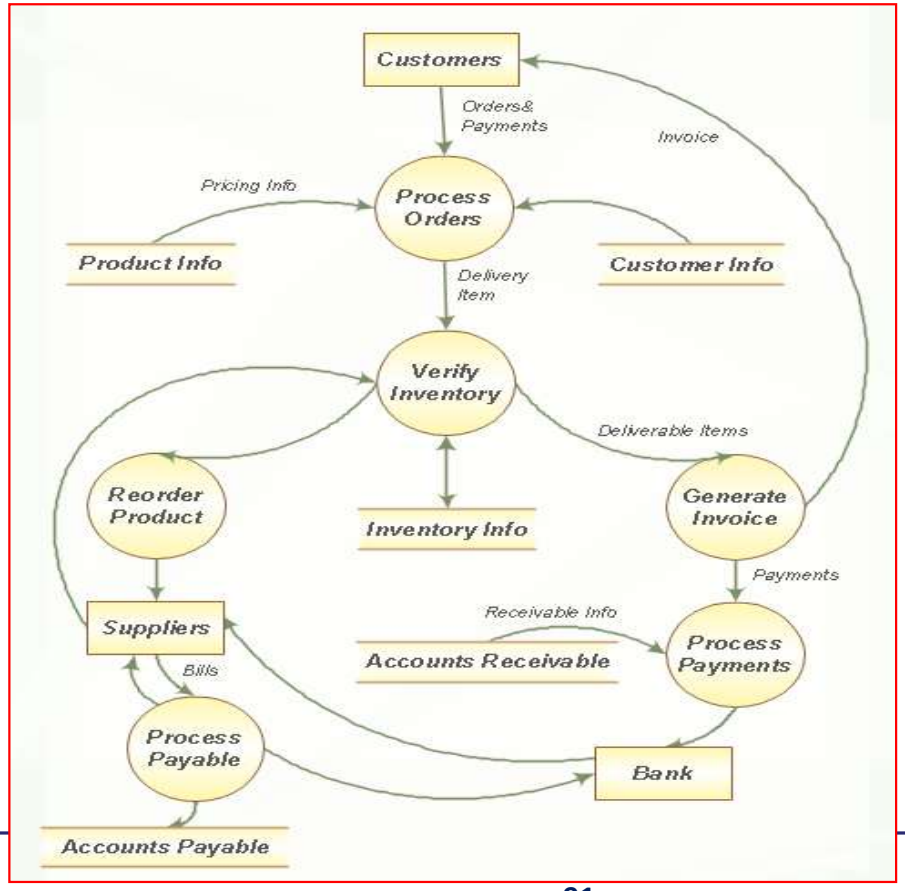
Example: Insurance Claims Software



DFD Styles

Coad and Yourdon Diagrams (the Yourdon symbols)

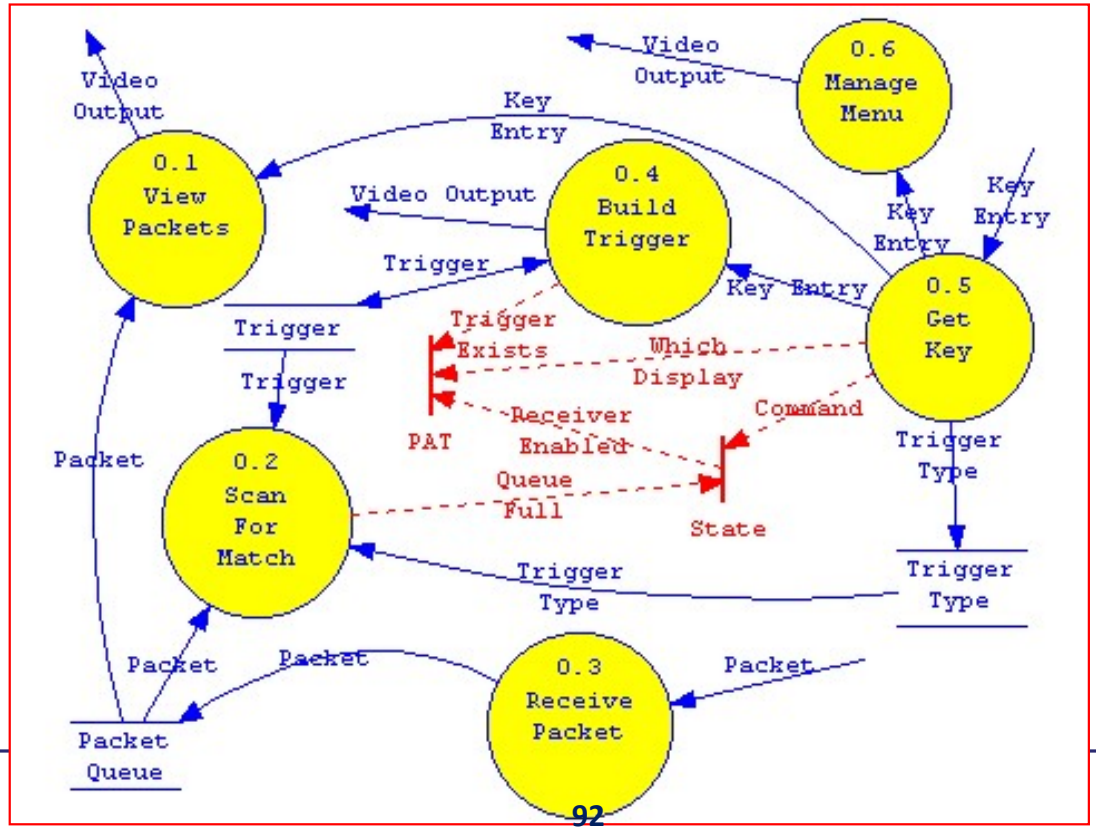
Example: Verify Inventory



DFD Styles

Yourdon & DeMarco Diagrams

A Yourdon & DeMarco style DFD is shown below. It includes both data and control flow as required in the Hatley/Pirbhai method typically used in real-time system analysis and design.



Software

DFD software

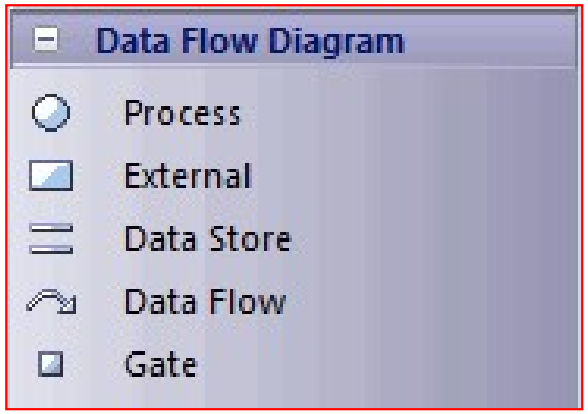
Enterprise Architect

http://www.sparxsystems.com/

- Data Flow Diagram Toolbox Page

  You can access the Data Flow Diagram page of the Toolbox through the More tools | Data Flow Diagrams menu option.

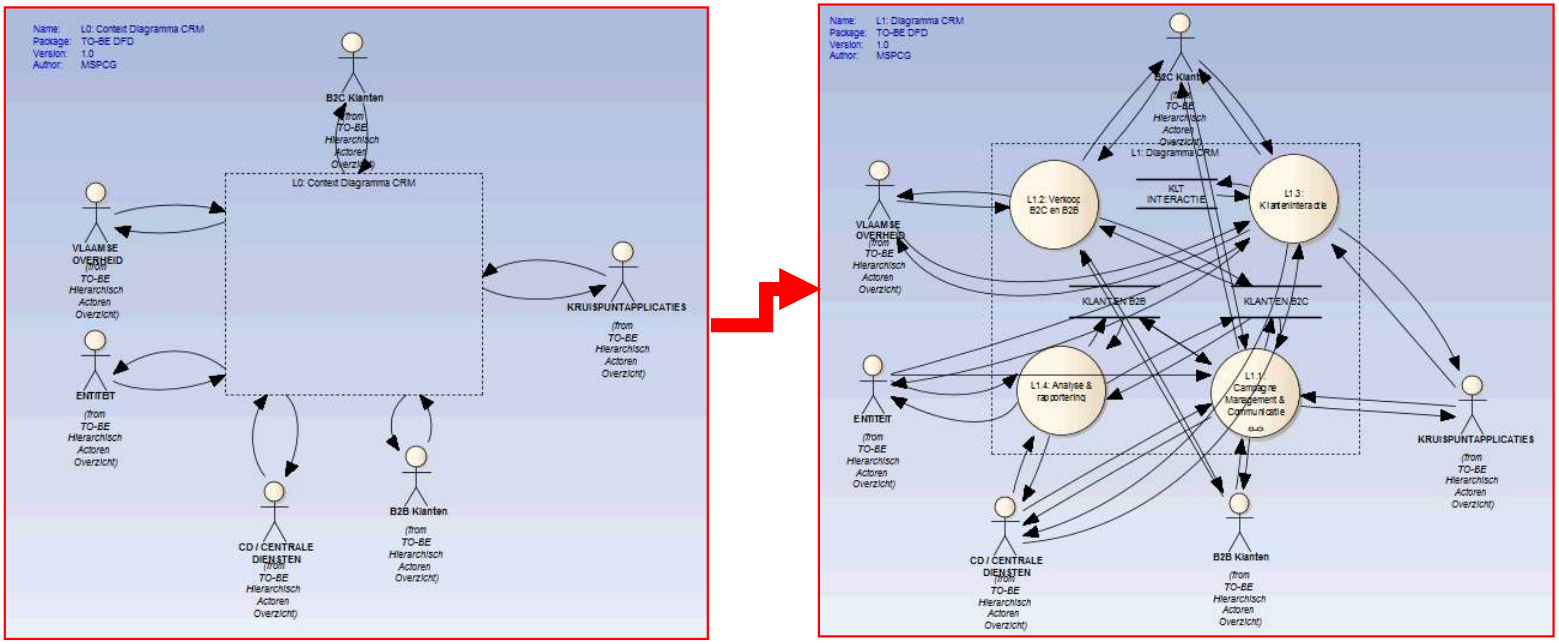
  The following icons are available:



Case studies

Project CRM / Company

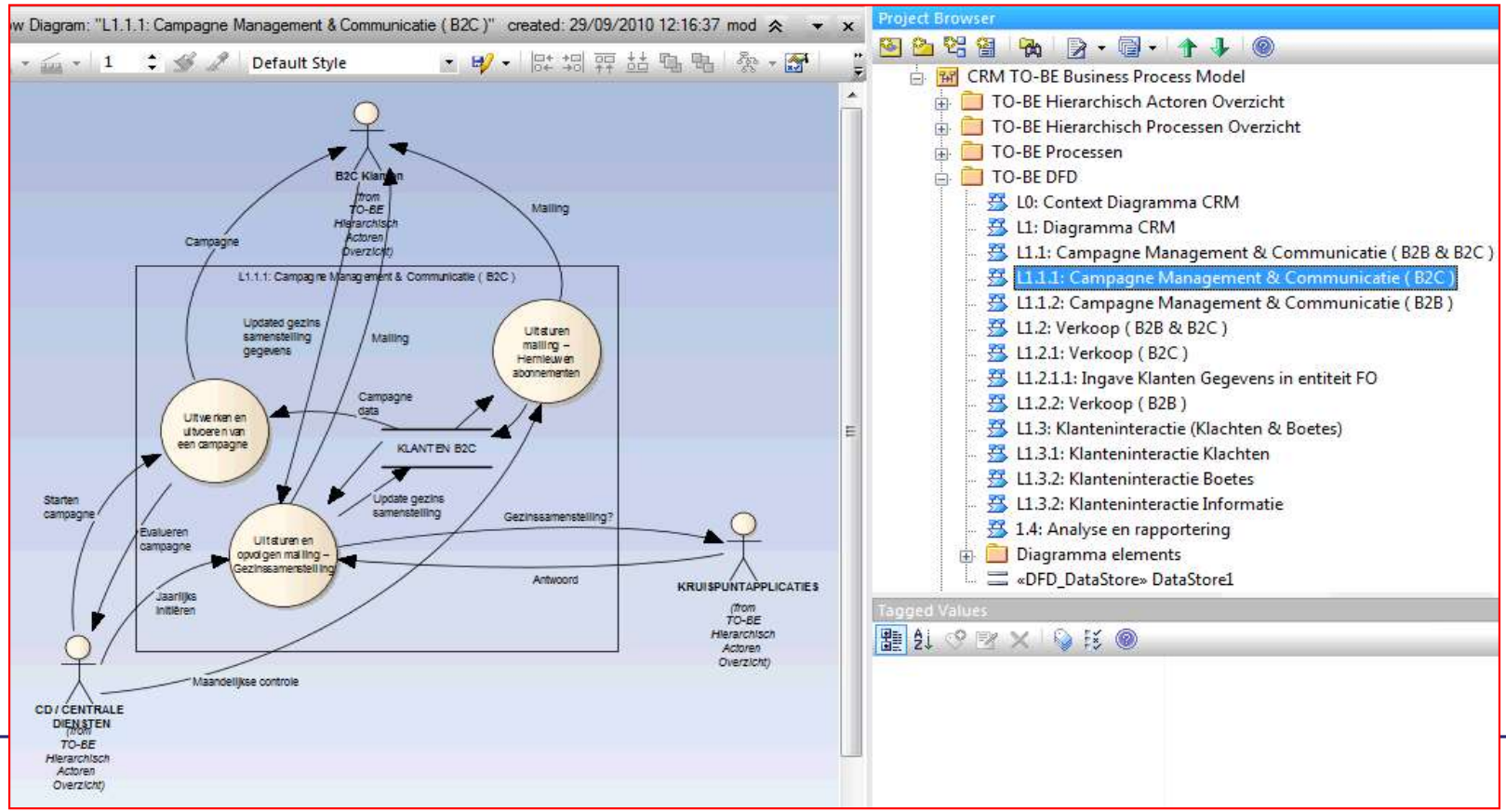
DFD Analysis (Yourdon symbols) of the new CRM system of Company.



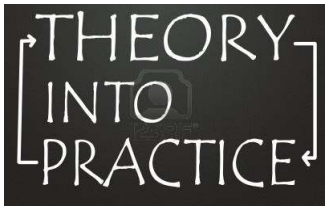
Case studies

Project CRM / Company

Via Navigation ..



Data Flow Diagram - Exercise

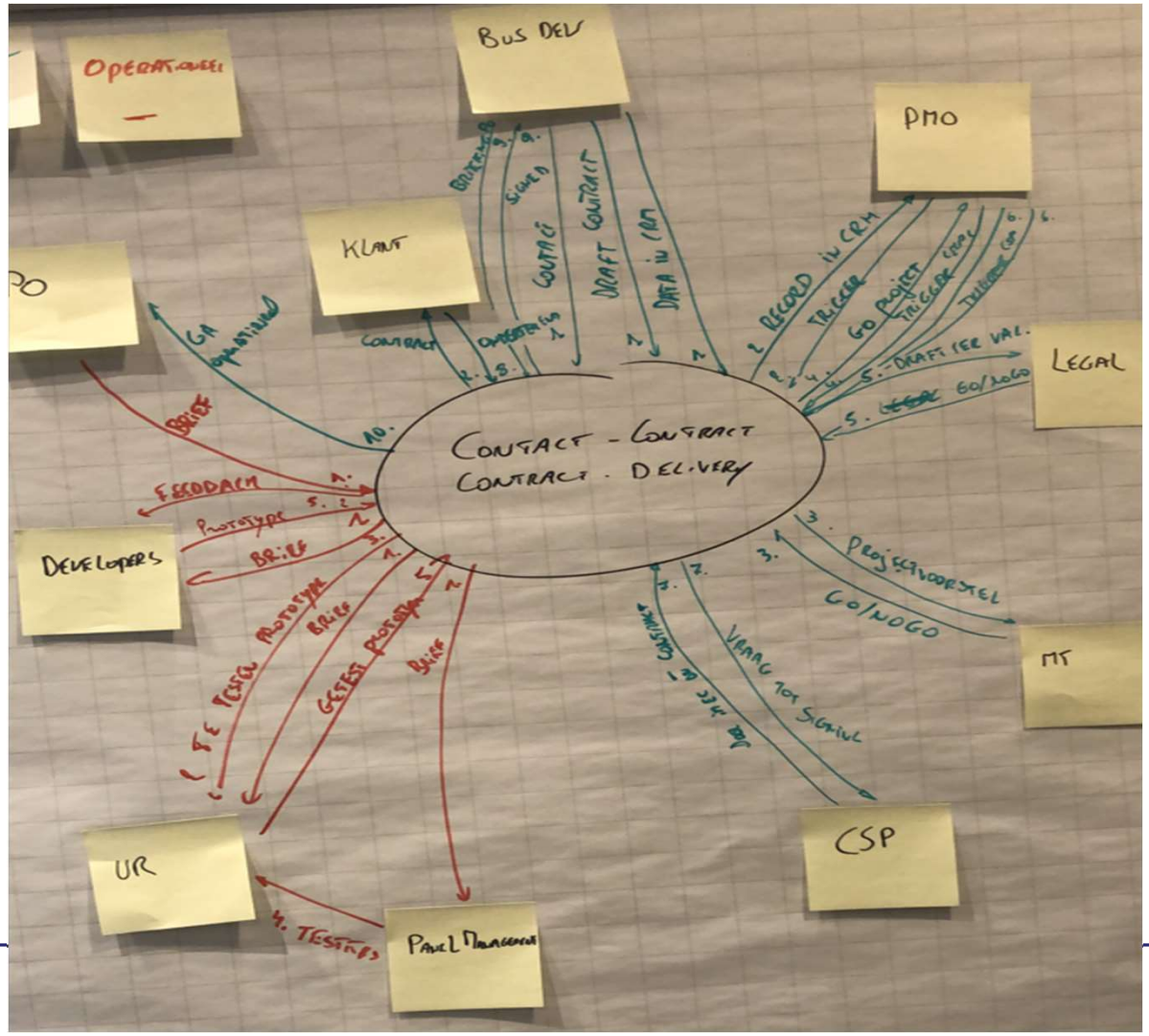


Exercise: Make a Data Flow Diagram

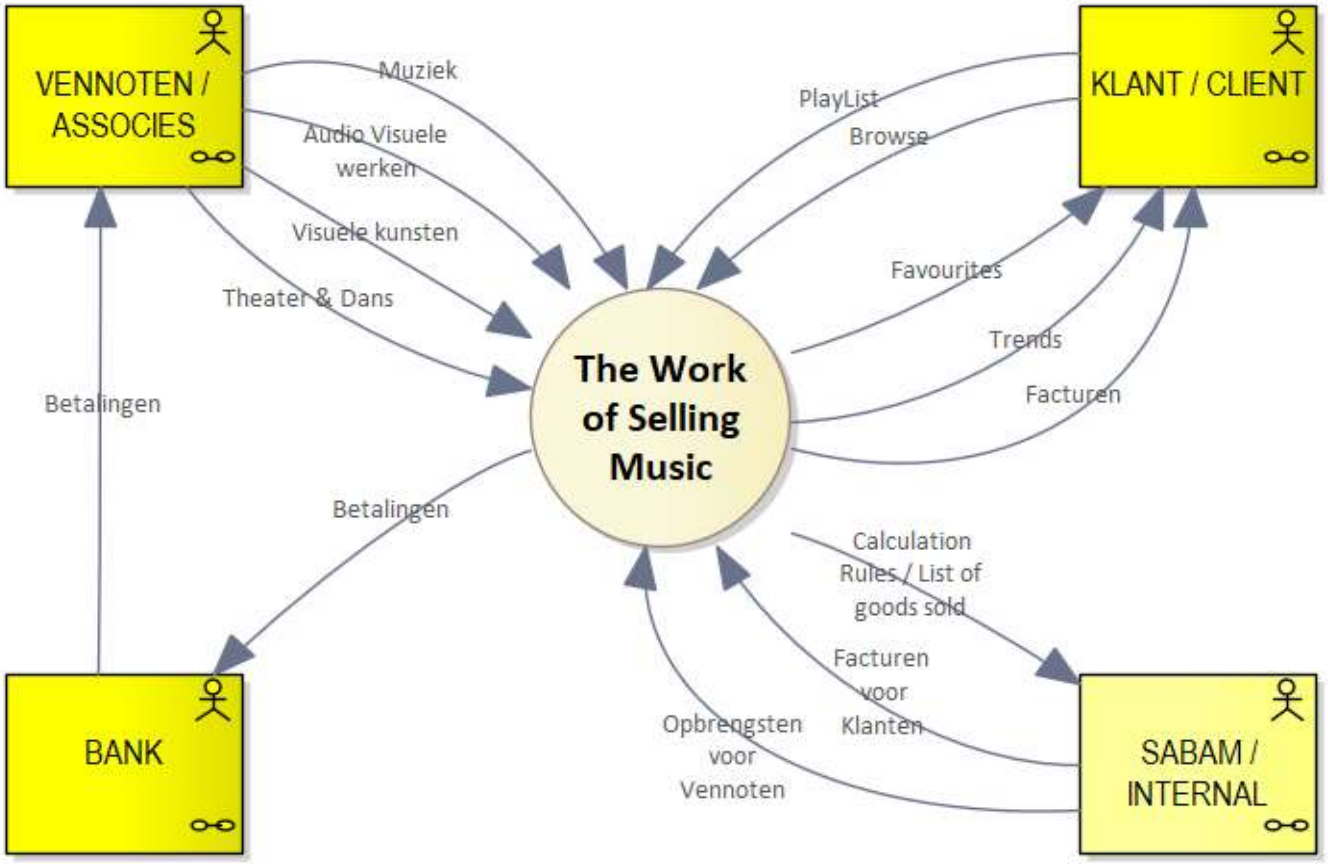
- Make a Data Flow Diagram of an in-house software system and their stakeholders. Model Level 1 (Context Diagram) and Level 2.

- Present your Data Flow Diagram

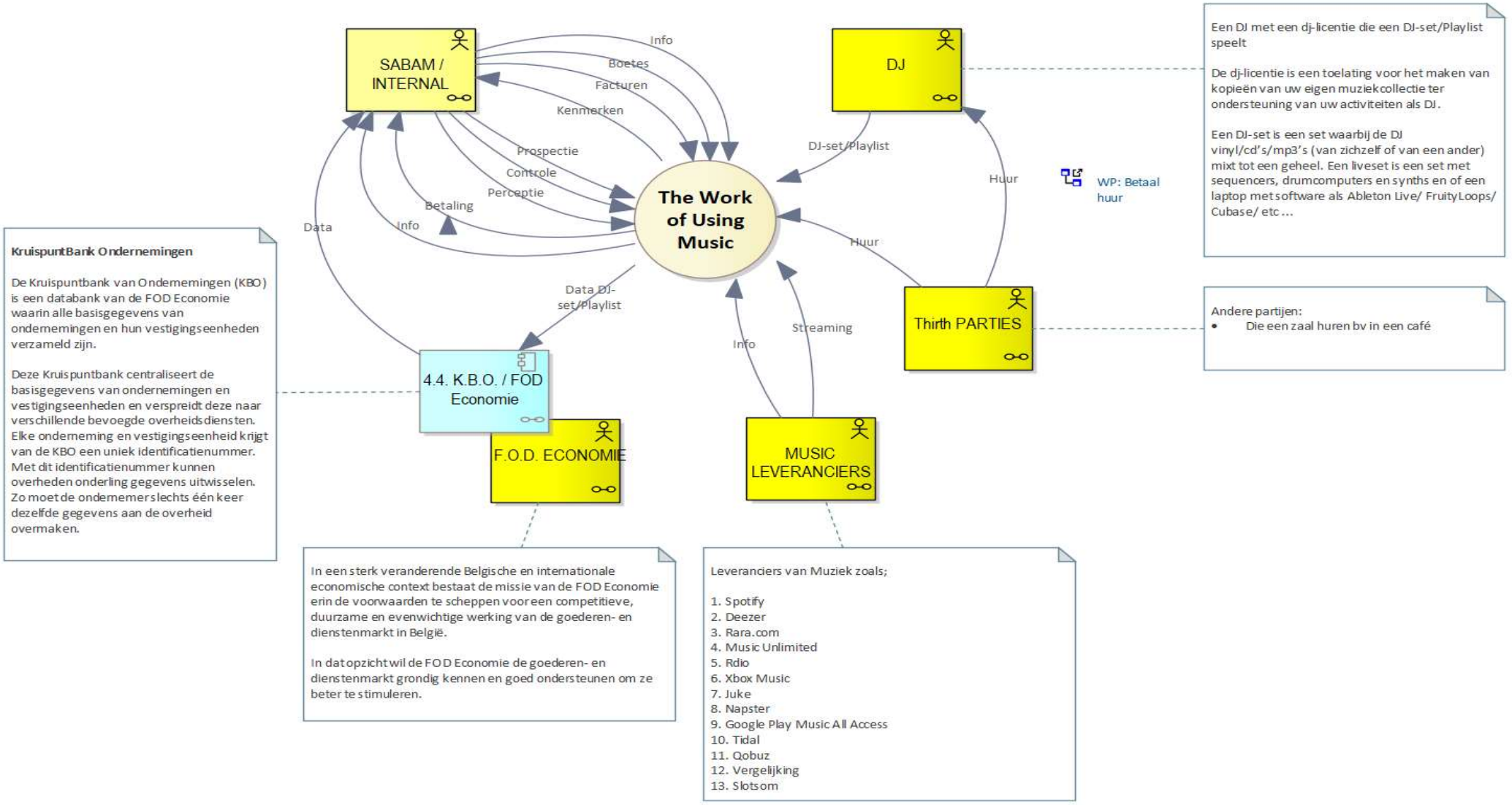
Data Flow Diagram - Examples



Data Flow Diagram - Examples



Data Flow Diagram - Examples



Part VII: Process models in BPMN

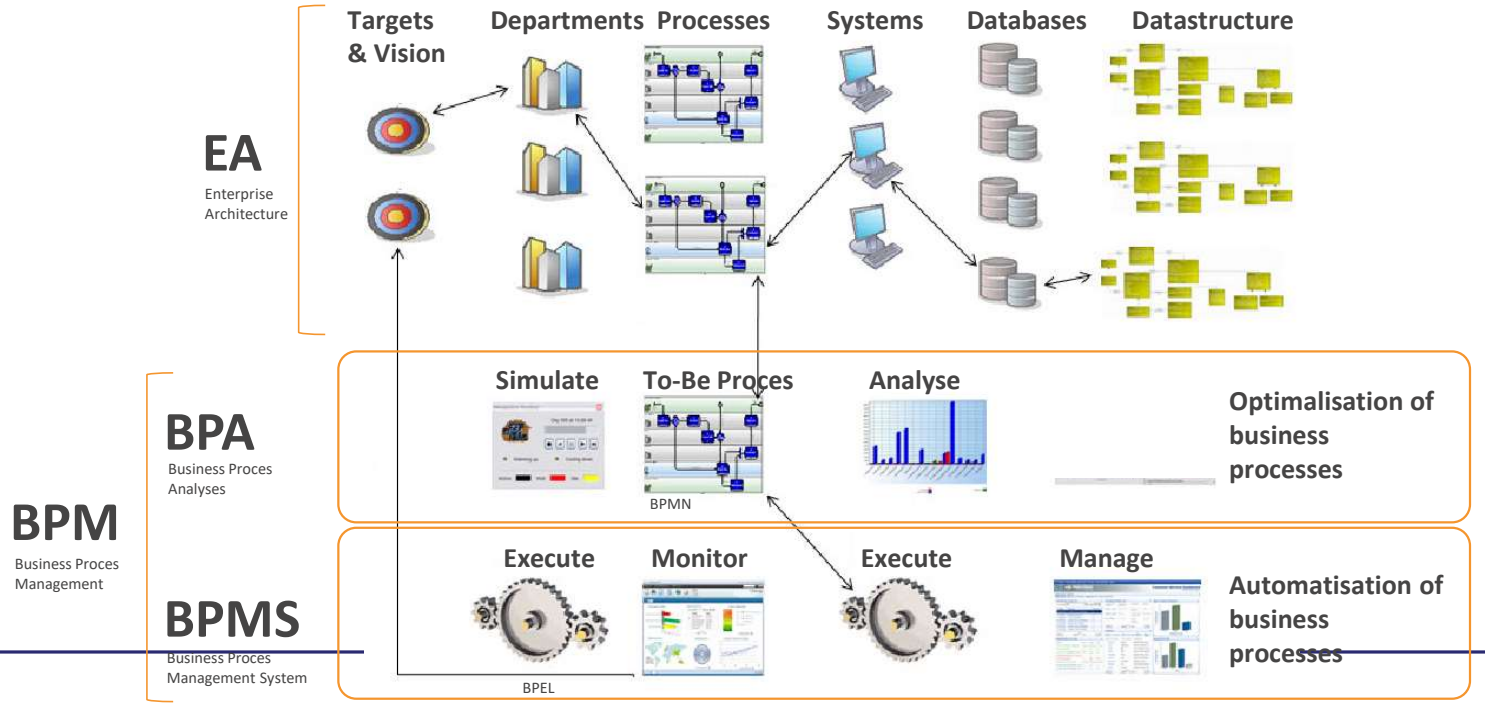
- Positioning
  - Business Architecture
- Busines analysis versus Functionale analysis
- Principles
- What is a process
- BPMN Notations
  - Model the actors and data-objects
  - Model the activities and events within a business process
- BPMN Styles
- Software
  - On the market
- Case studies
  - Real case examples

- Demo
- Exercise

Positioning

Enterprise Architecture

An **EA** / Enterprise Architecture framework such as EA³ Cube, TOGAF or Zachman is a tool to align Strategy, Business and Technology ("EA = S + B + T"). It is built up from many different artifacts, such as the Strategy Plan, SWOT analysis, Operating Scenarios, Process Models, Data Entity Schemas, Data Models, Business Use Cases, Investment Business Cases, Workforce plans, ...

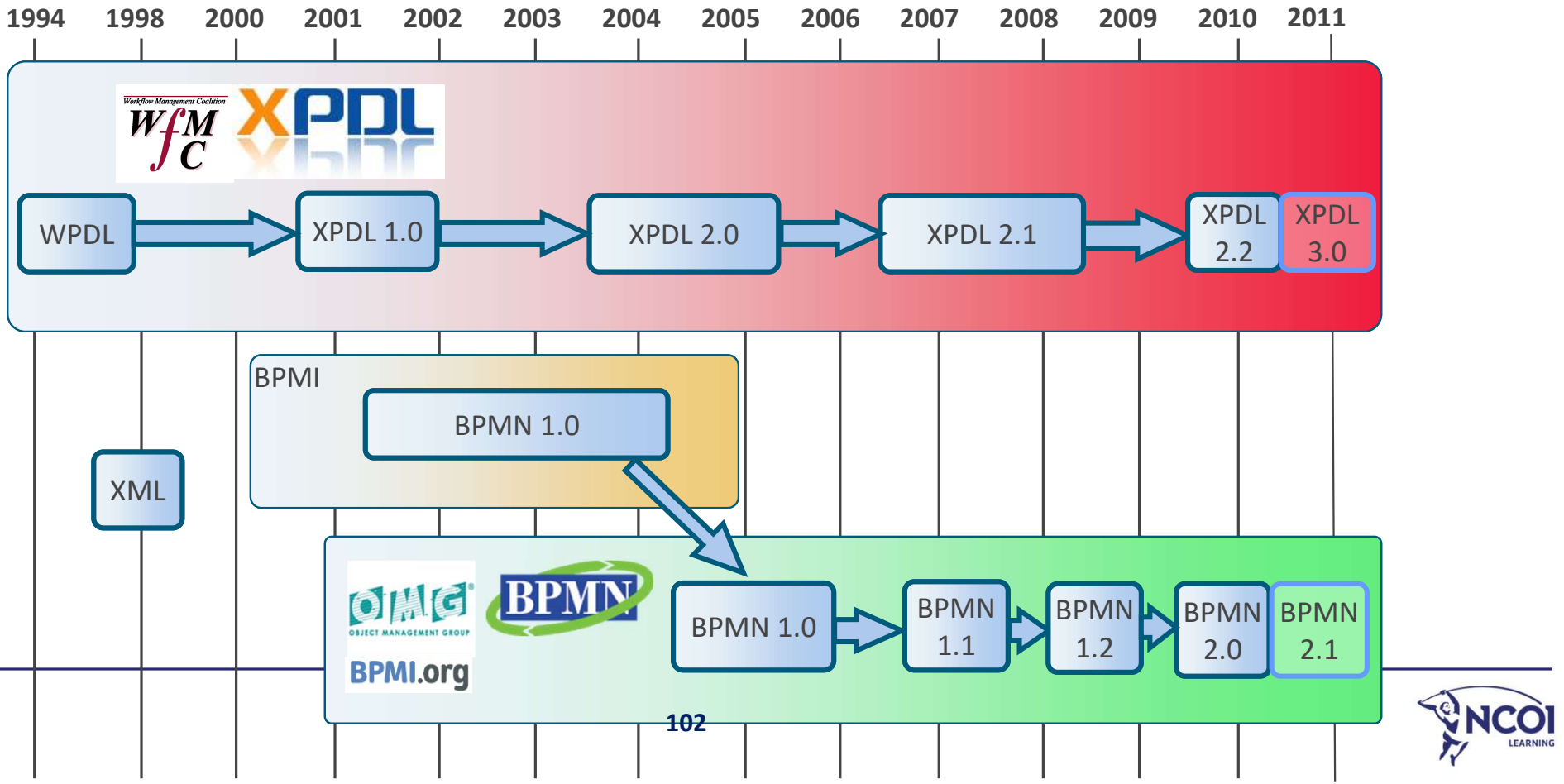


Positioning

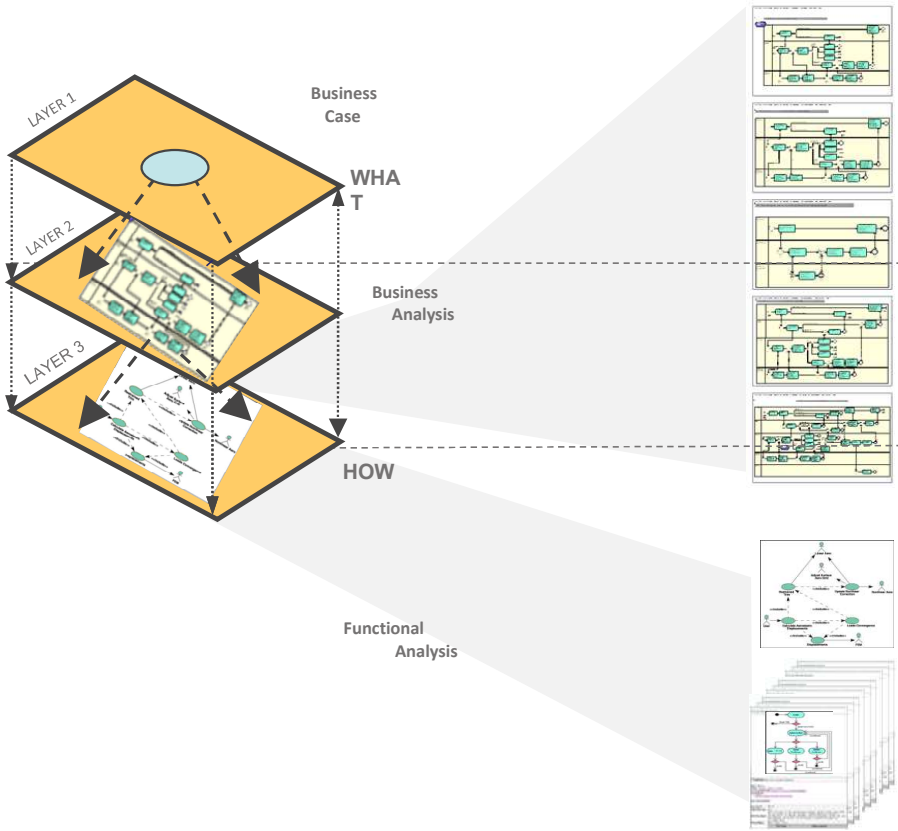
BPMN

http://www.wfmc.org/xpdl.html
http://www.omg.org/bpmn/

About BPMN – The History



Business Analysis vs. Functional Analysis



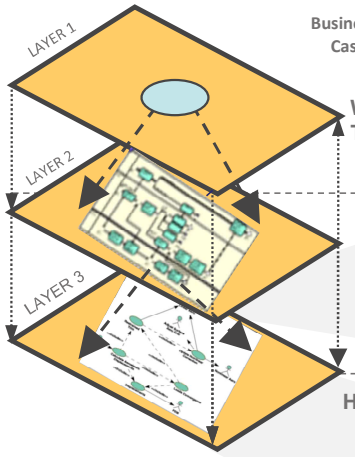
Business analysis

- Start& stop events
- Sequence of activities
- Connectors
- Main exceptions

Functional analysis

- Actors
- Purpose
- Pre-condition
- Post-condition
- Assumptions
- Triggers
- Activity scheme
- Activity steps – alternates / exceptions in detail
- (User-Interface)
- Business-rules
- Performance en freq.
- CRUD table

Business Analysis vs. Functional Analysis



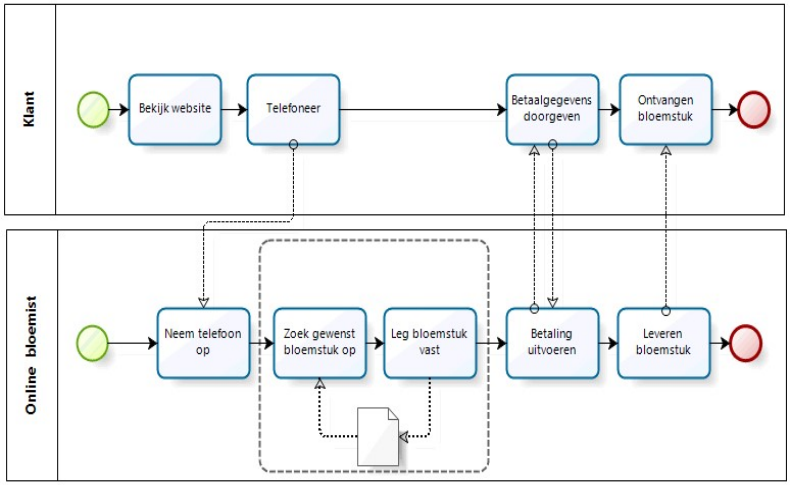
Business Case

WHAT

Business Analysis

HOW

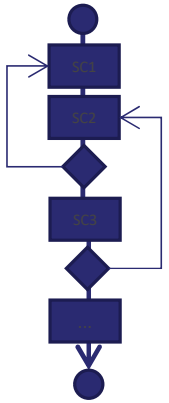
Functional Analysis



To be automated

- Actors
  - Klant
  - Online bloemist
- Purpose
  - Opzoeken, selecteren en bestellen bloemstuk
- Pre-condition
  - Selectie is gemaakt door klant
- Post-condition
  - Bloemstuk is vastgelegd
- Assumptions
- Triggers
- Activity scheme
- Activity steps – alternates / exceptions in detail
- (User-Interface)

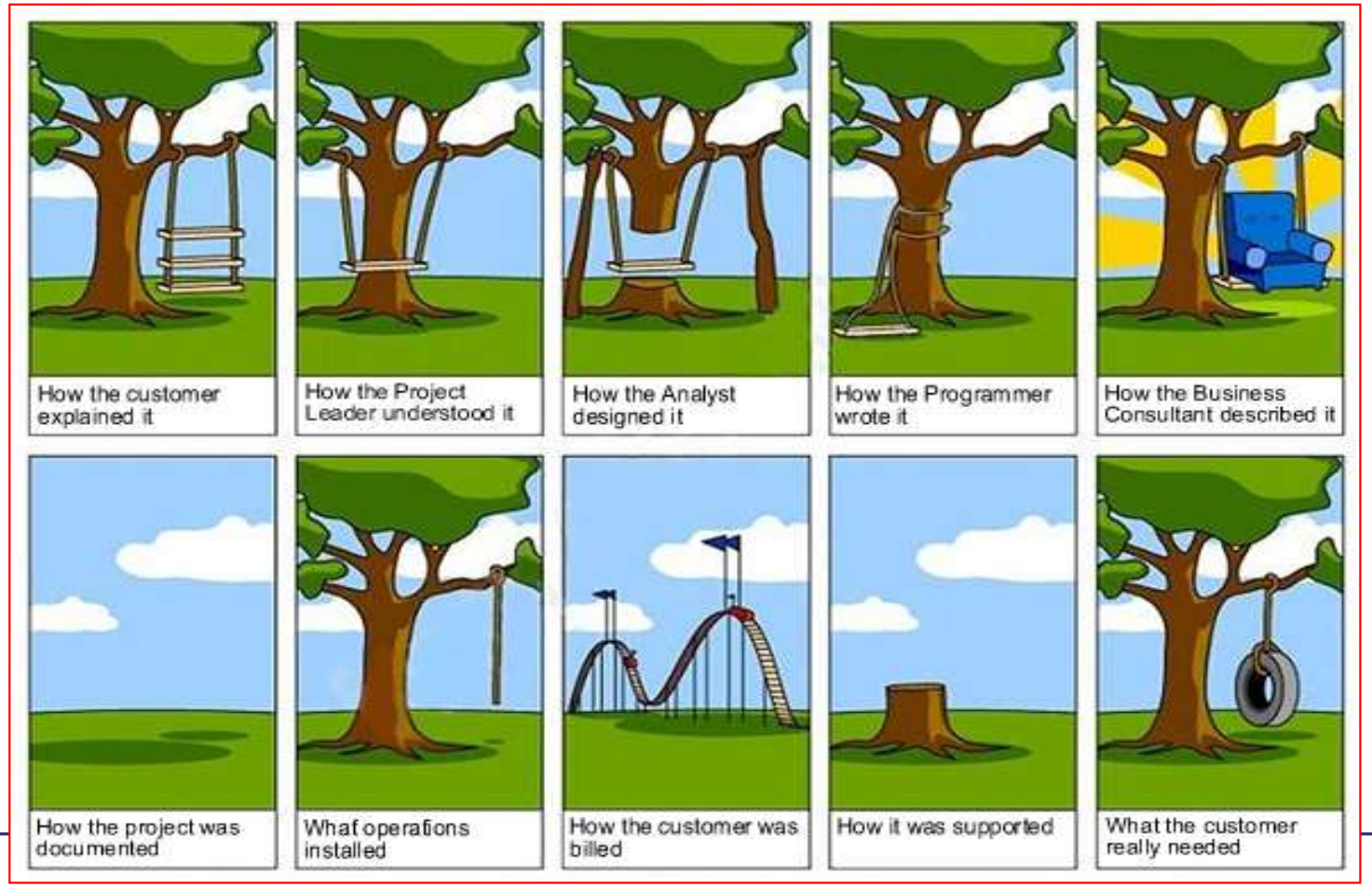
SC1: Homepage SC2: Select type SC3: Select flower, options & delivery date



- Business-rules
  - BR01: Check type …
  - BR02: Check flower …
  - BR03: Check options & delivery date
- Performance en freq.
- CRUD table

Business Analysis vs. Functional Analysis

Why Business Analysis and Functional Analysis ?



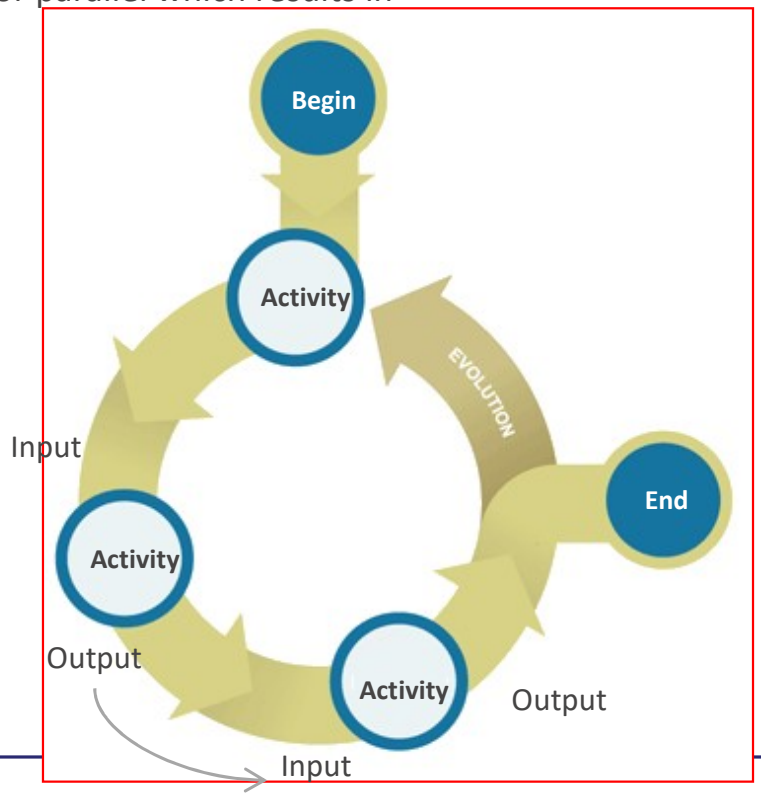
What is a process

Definition of a process

A process is:

A specific **ordering of work activities** sequential or parallel which results in

transformation that **creates value**
across time (continue / wait for)
and **place** with
a **beginning** (trigger)
an **end** and
clearly defined **inputs**
and **outputs** to achieve a
defined **business outcome**



BPMN Notations

About BPMN – BPMN consists of one Diagram.

BPMN consists of one diagram – called the **Business Process Diagram** (BPD).

The BPMN Business Process Diagram is a flow-chart based notation and has been designed to be easy to use and understand, but also provides the ability to model complex business processes.

BPMN Notations

Events

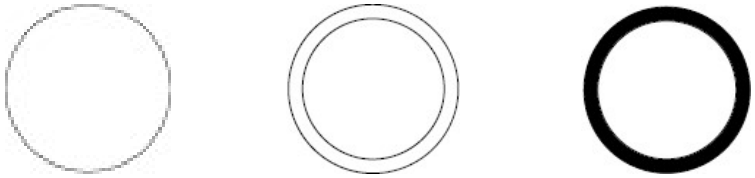
A BPD has a small set of (three) core elements, which are the Flow Objects, so that modelers do not have to learn and recognize a large number of different shapes.

The first Flow Object is:

An **Event** is represented by a circle and is something that "happens" during the course of a business process. These Events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow:

Start, Intermediate, and End

Event



BPMN Notations

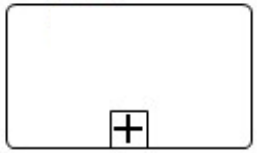
Activities

A BPD has a small set of (three) core elements, which are the Flow Objects, so that modelers do not have to learn and recognize a large number of different shapes.

The second Flow Object is:

An **Activity** is represented by a rounded-corner rectangle (see the figure to the right) and is a generic term for work that company performs.

An Activity can be atomic or non atomic (compound). The types of Activities are: **Task** and **Sub-Process**. The Sub-Process is distinguished by a small plus sign in the bottom center of the shape.

	Task	Sub-Process
Activity		

BPMN Notations

The (core) syntax of BPMN – 1.3 Flow Objects

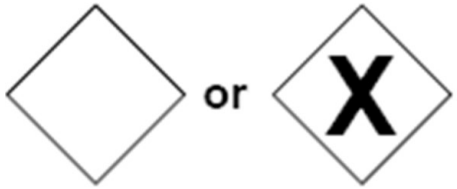
A BPD has a small set of (three) core elements, which are the Flow Objects, so that modelers do not have to learn and recognize a large number of different shapes.

The third Flow Objects is:

A **Gateway** is represented by the familiar diamond shape and is used to control the divergence and convergence of Sequence Flow. Thus, it will determine traditional decisions, as well as the forking, merging, and joining of paths. Internal Markers will indicate the type of behavior control.

Exclusive (XOR) Data-Based

Gateway

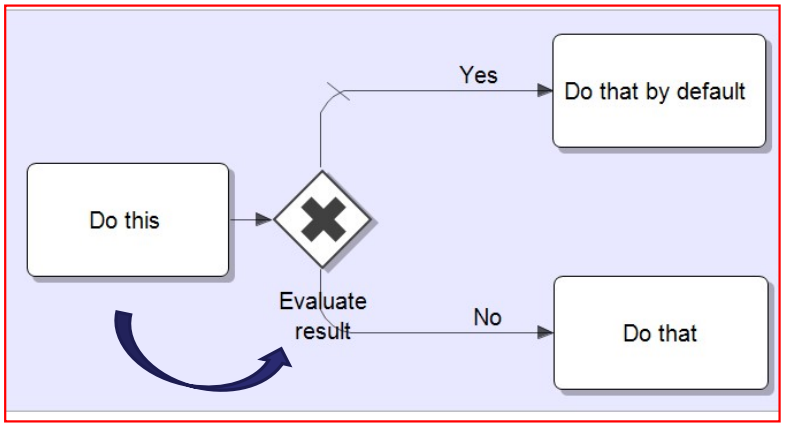


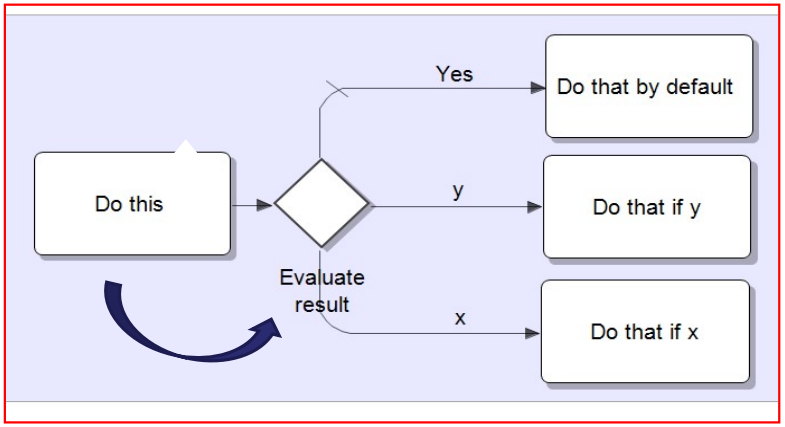
BPMN Notations

Examples

Example: Exclusive XOR gateway

Gateways can be shown with or without an internal ' **X** ' marker.





BPMN Notations

Pools

Many process modeling methodologies utilizes the concept of swim lanes as a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities. BPMN supports swim lanes with two main constructs.

The first type of BPD swimlane object is:

A **Pool** represents **a process / an organization** dependent on the style of modeling. It also acts as a graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.

Pool

BPMN Notations

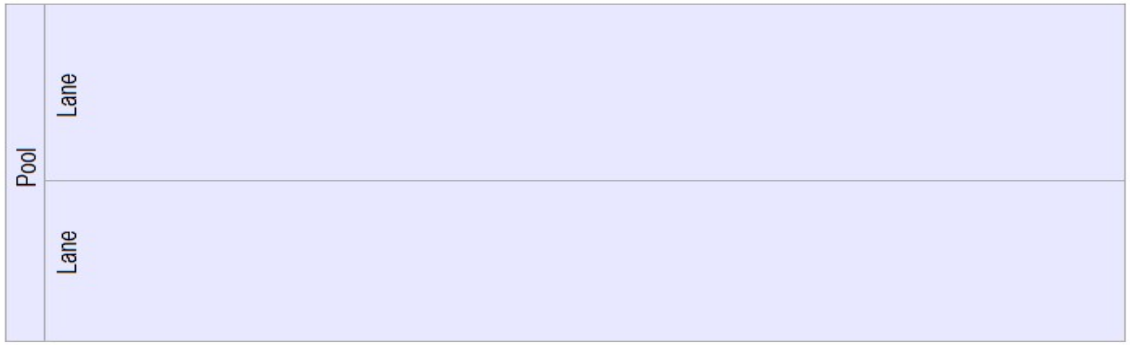
Swim lanes

Many process modeling methodologies utilizes the concept of swim lanes as a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities. BPMN supports swim lanes with two main constructs.

The second type of BPD swimlane object is:

A **Lane** represents **a department** within that process / organization (Pool) or a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organize and categorize activities. By taking processes and placing them in pools or lanes, you are specifying who does what; for events, you specify where they occur; and, for gateways, you specify where decisions are made, or who makes them.

(Swim) lanes

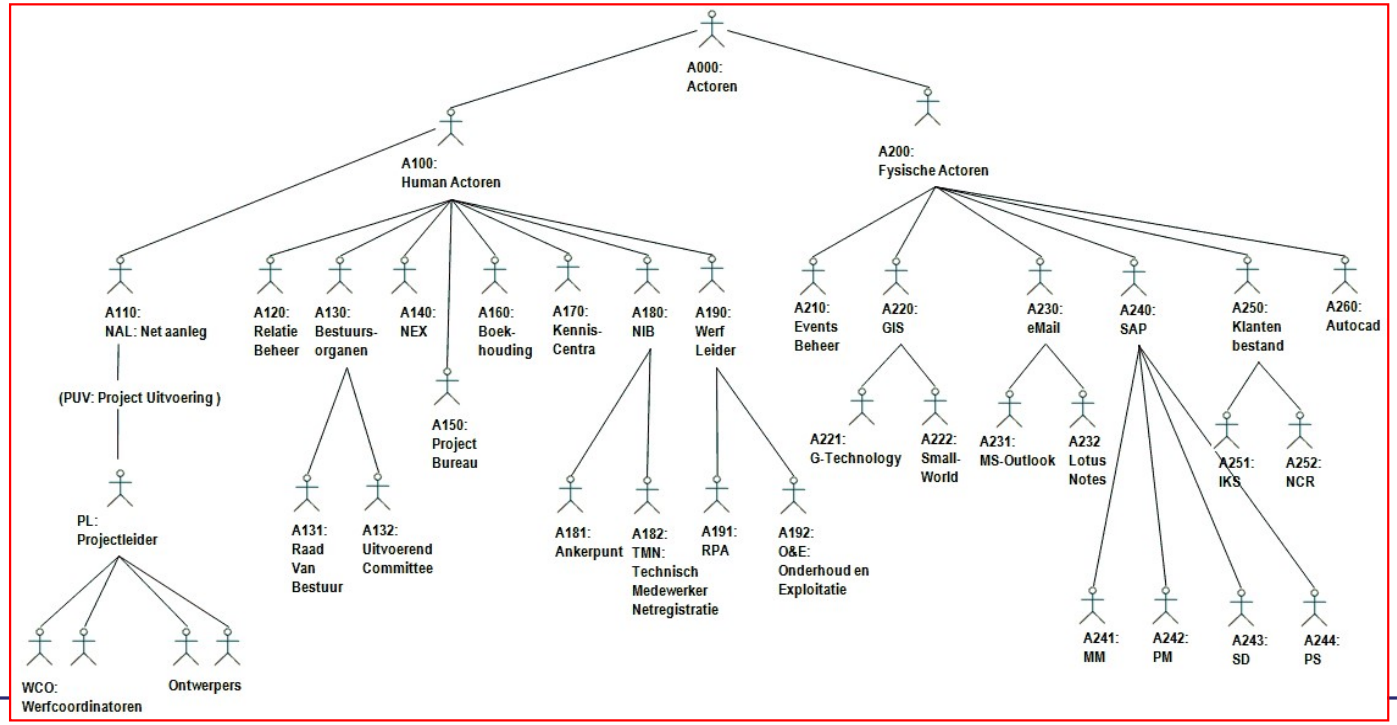


BPMN Notations

Actors

- Actors can be ranked (example numbers Axxx) in **Human** and **Physical** actors.

- Actors (Human and Physical) can be ranked by **Department / Team**.



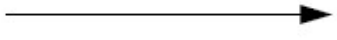
BPMN Notations

Sequence flow

The Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. There are three Connecting Objects that provide this function.

The first connector is:

A Sequence Flow is represented by a solid line with a solid arrowhead (see the figure to the right) and is used to show the order (the sequence) that activities will be performed in a Process. Note that the term “control flow” is generally not used in BPMN.

Sequence Flow 

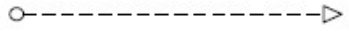
BPMN Notations

Message flow

The Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. There are three Connecting Objects that provide this function.

The second connector is:

A **Message Flow** is represented by a dashed line with an open arrowhead and is used to show the flow of messages between two separate Process Participants (business entities or business roles) that send and receive them. In BPMN, two separate Pools in the Diagram will represent the two Participants.

Message Flow 

BPMN Notations

Association flow

The Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. There are three Connecting Objects that provide this function.

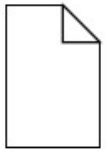
The third connector is:

An **Association** is represented by a dotted line with a line arrowhead and is used to associate data, text, and other Artifacts with flow objects. Associations are used to show the inputs and outputs of activities.

Association flow 

BPMN Notations

Artifact Data Object

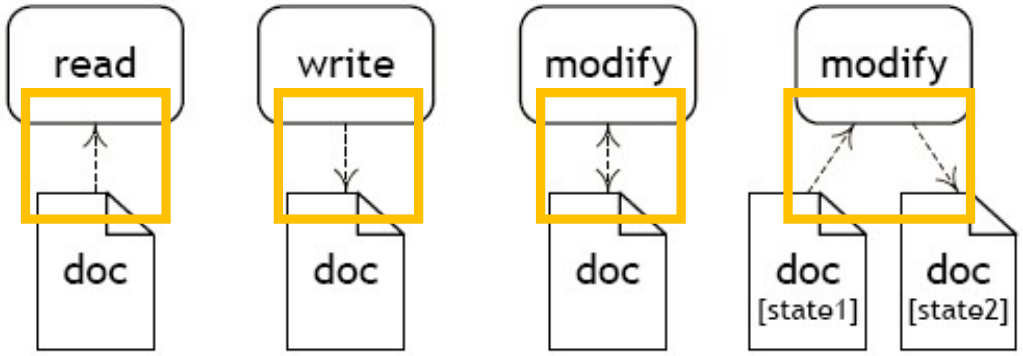


A **Data Object** represents information flowing through the process, such as business documents, e-mails or letters.

Attaching a data object with an **Undirected Association** to a sequence flow indicates hand-over of information between the activities involved.

A **Directed Association** indicates information flow. A data object can be read at the start of an activity or written upon completion.

A **Bidirected Association** indicates that the data object is modified, i.e. read and written during the execution of an actvity.



BPMN Notations

Artifact Data Store

The DataStore is introduced to represent persistent data shared by processes.

The Data Store is external to the process but has store/retrieve access to it.
A Data Store provides a place in the Process where Activities can retrieve or update stored information that will persist beyond the scope of the Process.
The Data Store uses directional data association to show data flow to/from the process.

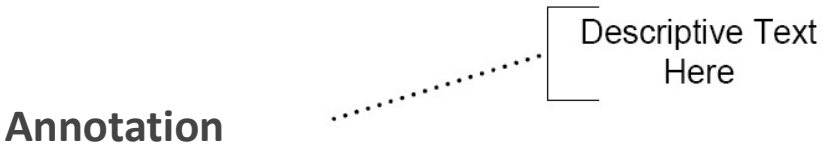
BPMN Notations

Annotation

BPMN was designed to allow modelers and modeling tools some flexibility in extending the basic notation and in providing the ability to additional context appropriate to a specific modeling situation, such as for a vertical market (e.g., insurance or banking). Any number of Artifacts can be added to a diagram as appropriate for the context of the business processes being modeled.

The current version of the BPMN specification pre-defines only three types of BPD Artifacts, the third is:

Annotations are a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram.



BPMN Notations

Group

BPMN was designed to allow modelers and modeling tools some flexibility in extending the basic notation and in providing the ability to additional context appropriate to a specific modeling situation, such as for a vertical market (e.g., insurance or banking). Any number of Artifacts can be added to a diagram as appropriate for the context of the business processes being modeled.

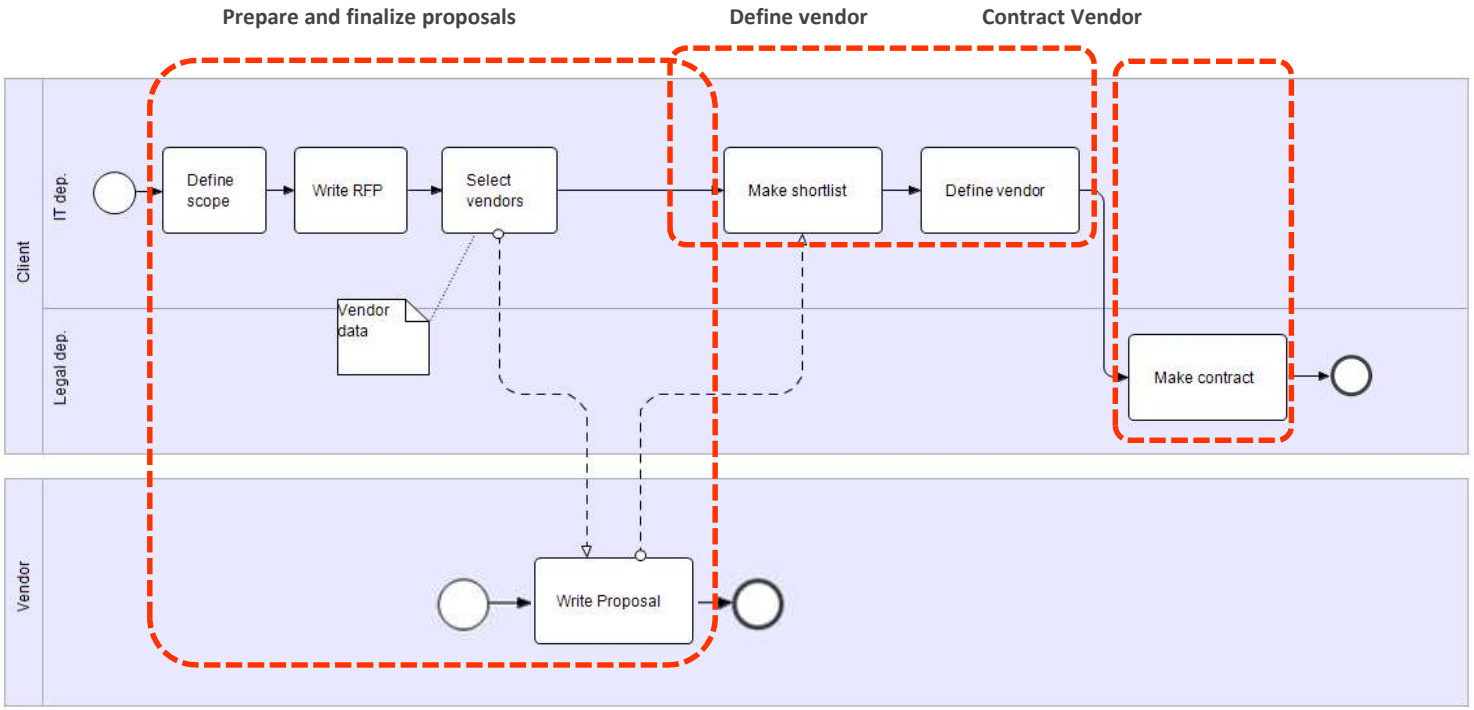
The current version of the BPMN specification pre-defines only three types of BPD Artifacts, the second is:

A **Group** is represented by a rounded corner rectangle drawn with a dashed line. The grouping can be used for documentation or analysis purposes, but does not affect the Sequence Flow.

Group

BPMN Notations

Example



BPMN Notations

Black-Box

Abstract (Public) Processes (also called black-box)

This represents the interactions between a private business process and another process or participant.

Only those activities that are used to communicate outside the private business process, plus the appropriate flow control mechanisms, are included in the abstract process.

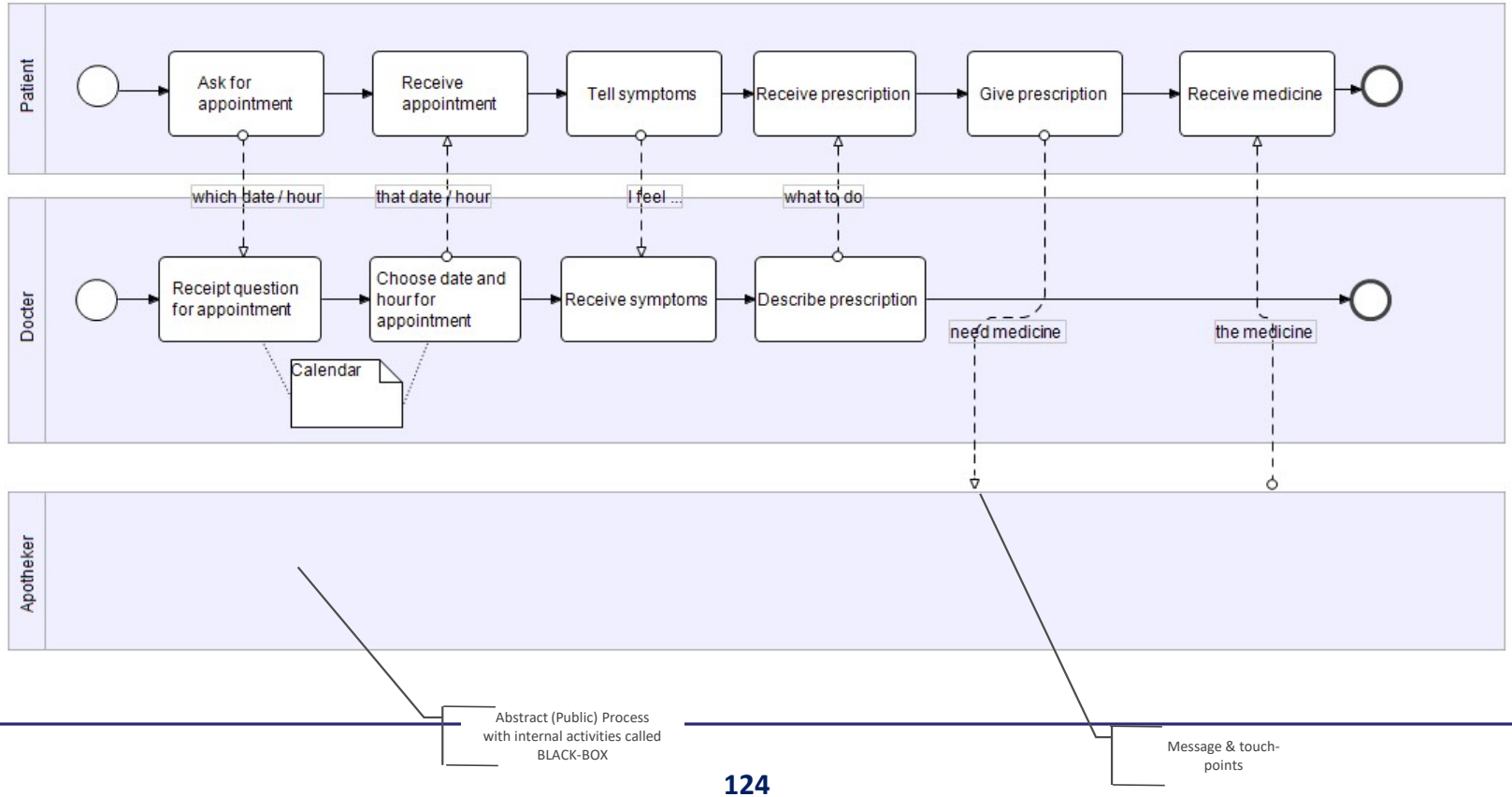
All other "**internal**" activities of the private business process are not shown in the abstract process.

Thus, the abstract process shows to the outside world the sequence of messages that are required to interact with that business process.

BPMN Notations

Black-Box - Example

Example: Abstract (Public) Processes (also called black-box)



BPMN Notations

Specialisations

A Message event:

A message arrives from a participant and triggers the start of the Process.

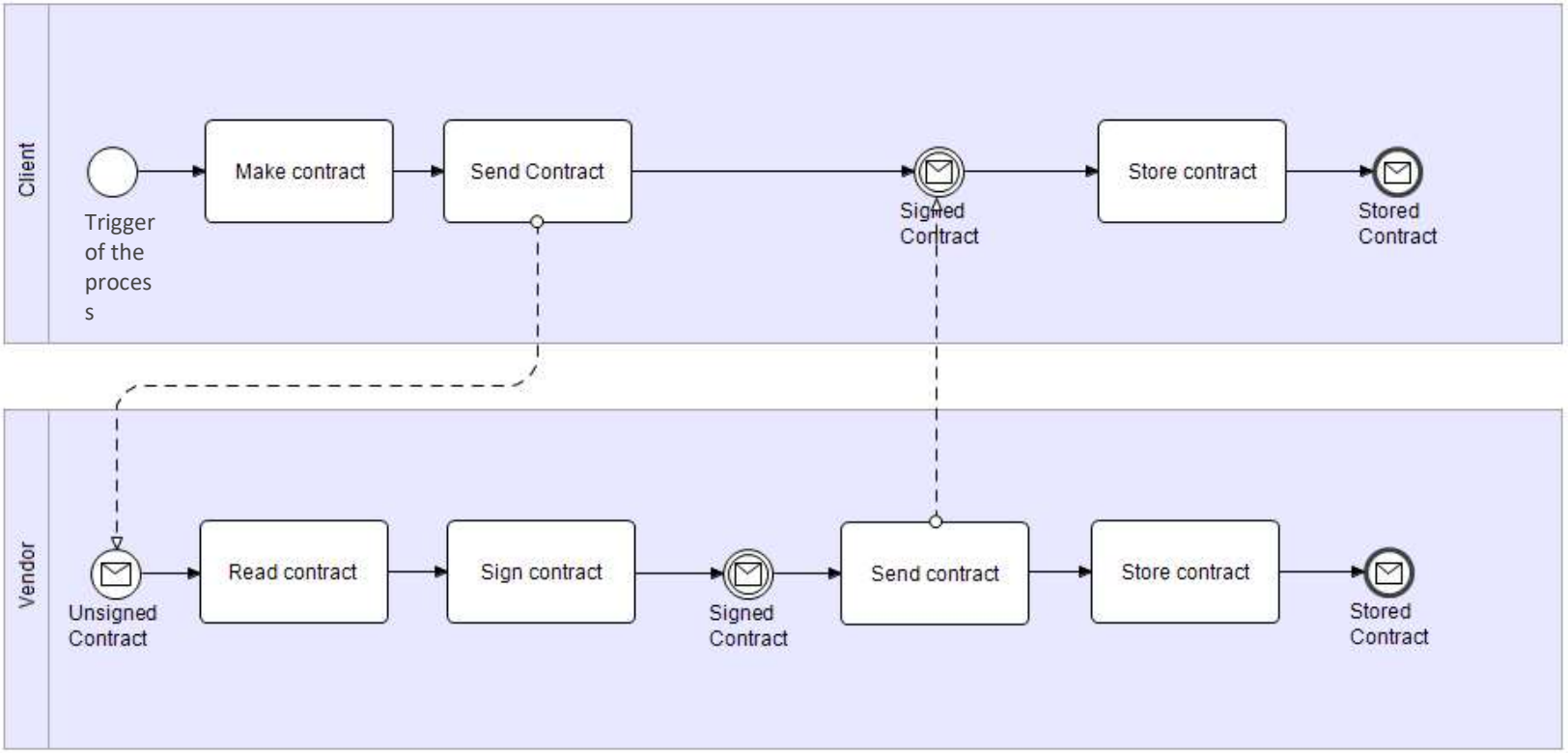
A message arrives from a participant and triggers the Event. This causes the Process to continue if it was waiting for the message, or changes the flow for exception handling.

This type of End indicates that a message is sent to a participant at the conclusion of the Process.

BPMN Notations

Message event - Example

Example: Message events



Software

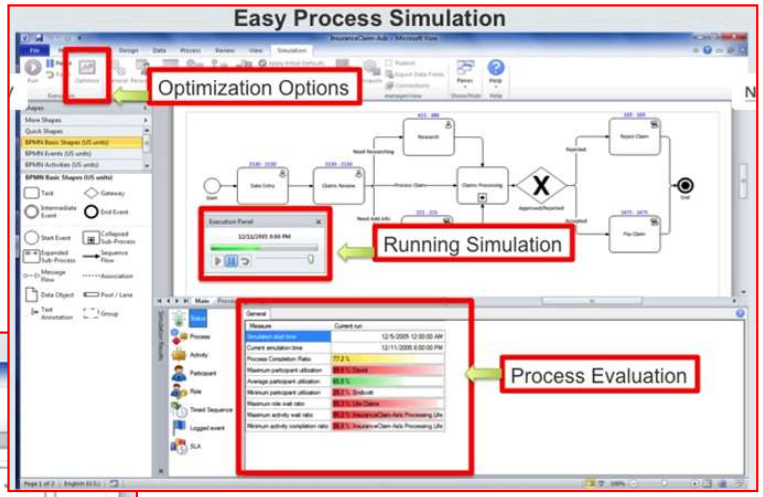
BPMN software

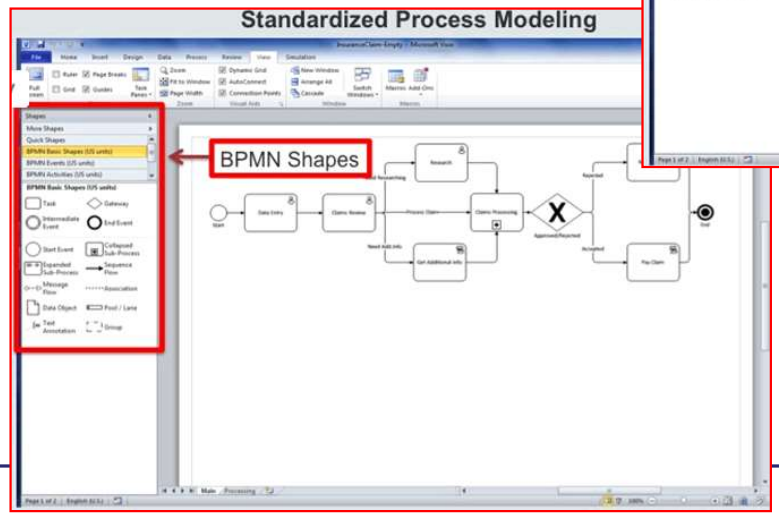
Microsoft Global 360

http://emea.microsoftstore.com/nl/nl-NL/Microsoft/Visio-Premium-2010

VISIO Premium 2010 & AnalystView 3.0 :
a stencil and simulation set for Microsoft Visio

- 100% BPMN
- BPMN 1.2 Syntax validation
- Proces simulation
- Executable processes
- Connection with SharePoint 2010





Software

Proces / Project software

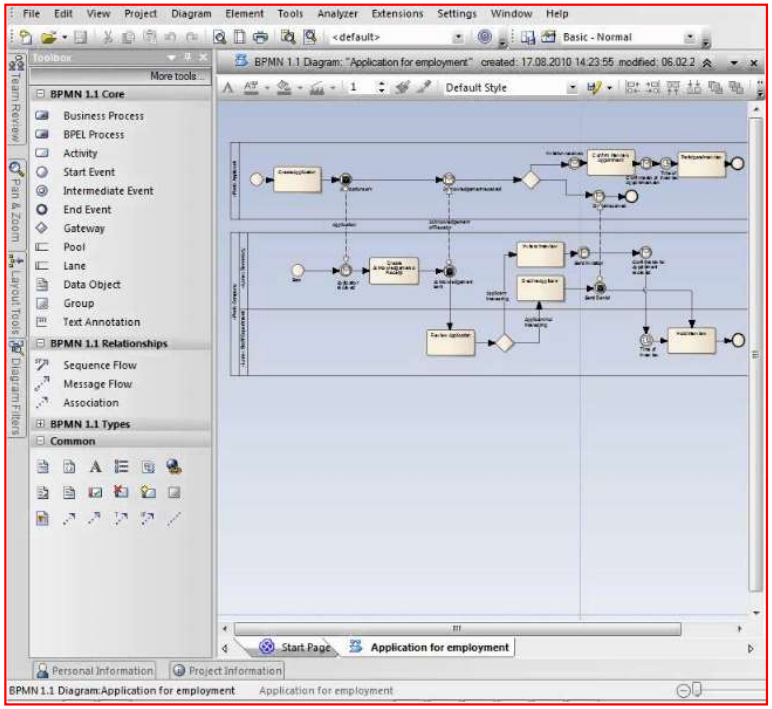
Enterprise Architect

- Enterprise Architect 15.x / 16.x

40 notation formats

- TOGAF
- ZACHMAN
- BPMN 1.2
- BPMN 2.0
- UML 2.0
- DFD
- ARCHIMATE 2.0
- MindMapping
- Screen design
- ...

LOW PRICE !



Software

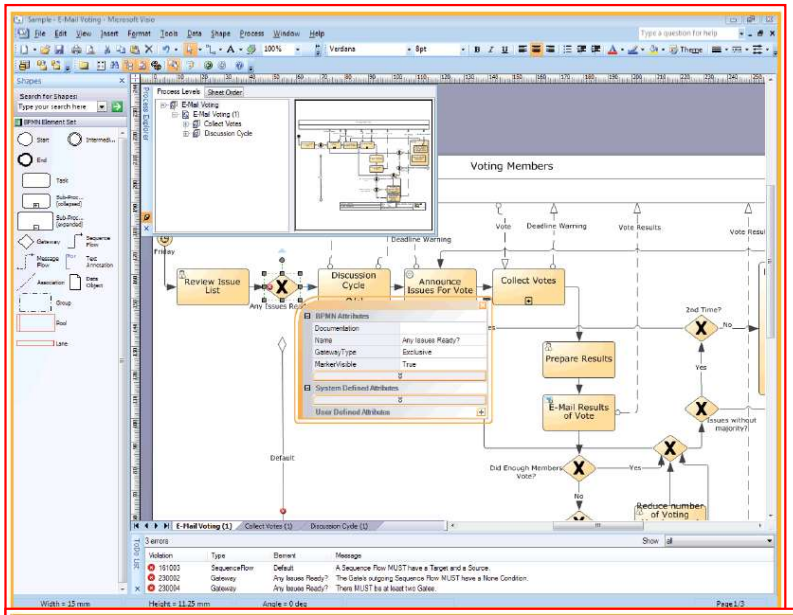
BPMN software

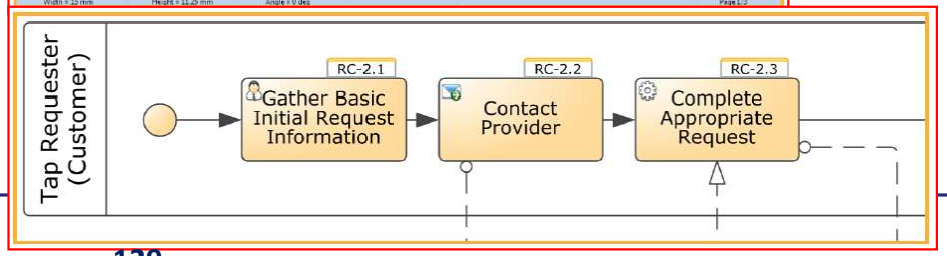
http://www.itp-commerce.com/

http://twitter.com/#!/itpcommerce

PROCES MODELER: a stencil and simulation set for Microsoft Visio

- 100% BPMN
- BPMN 2.0 Syntax validation





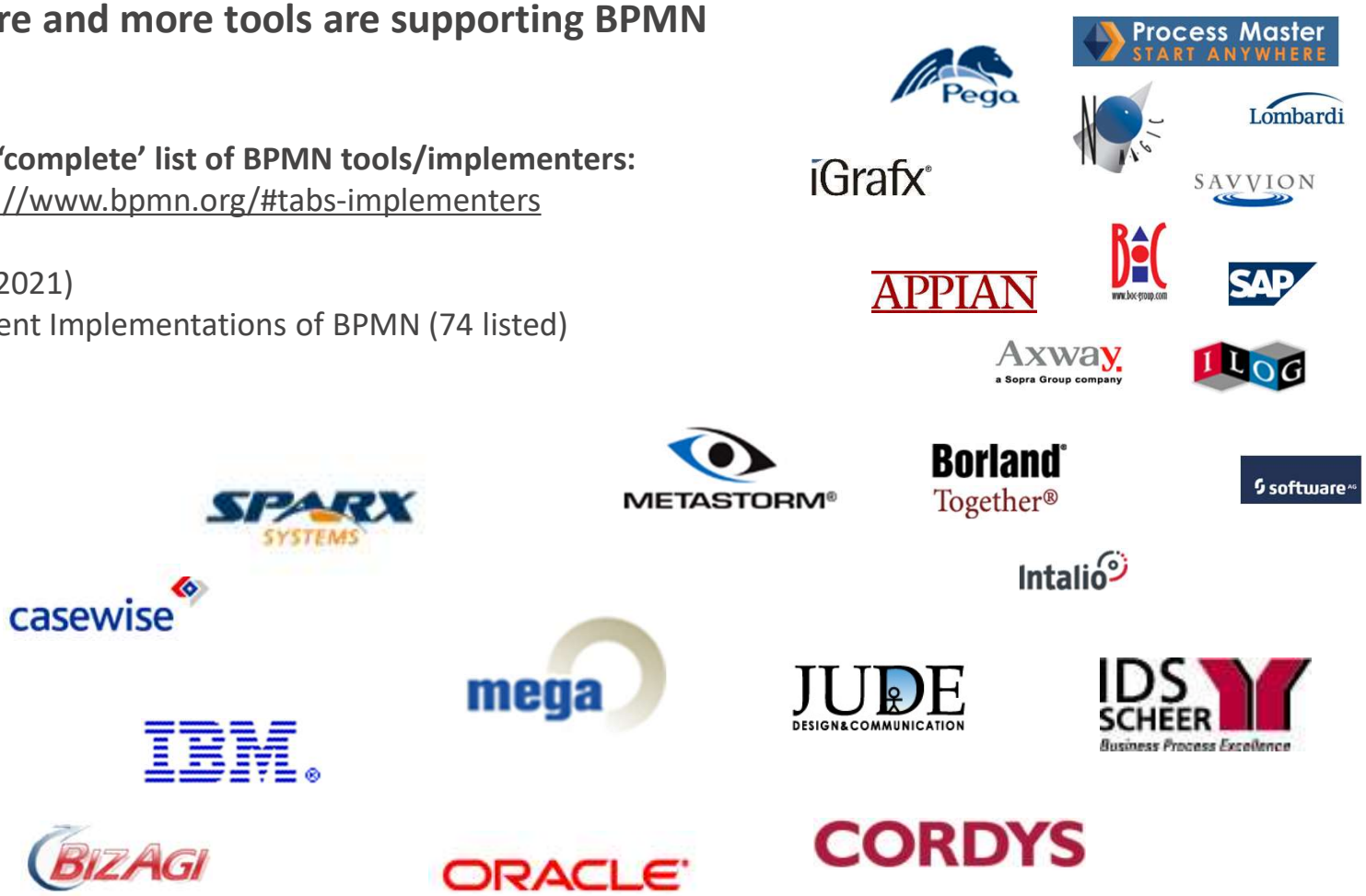
- Element numbering
- Documentation generation

Software

More and more tools are supporting BPMN

The 'complete' list of BPMN tools/implementers:
http://www.bpmn.org/#tabs-implementers

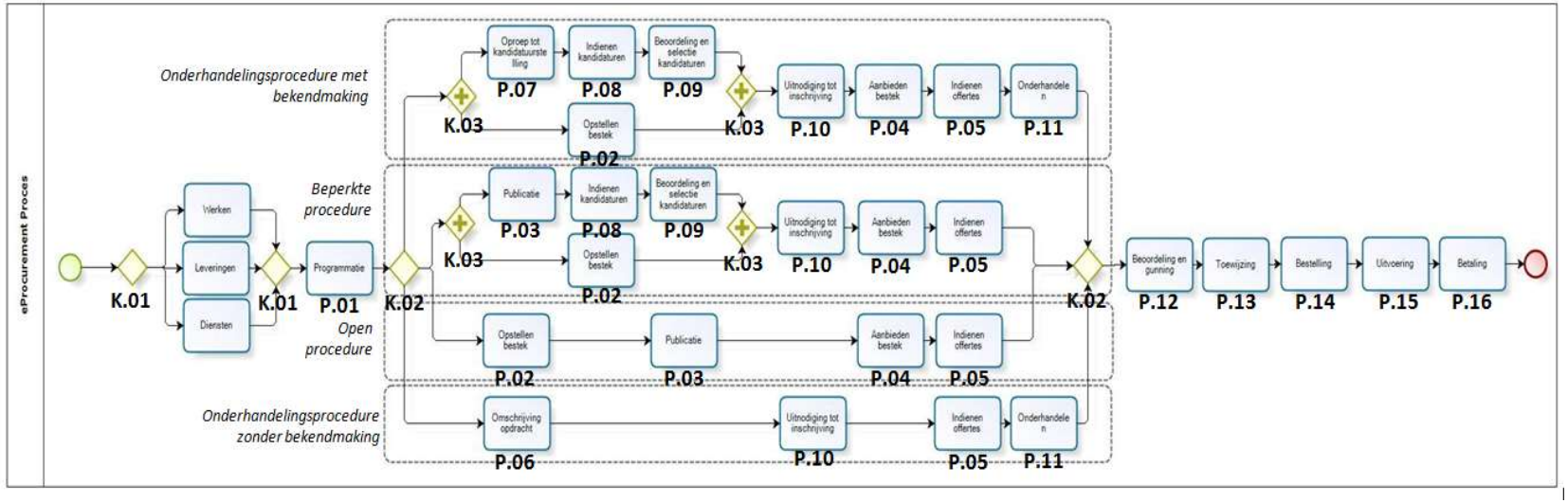
(11/2021)
Current Implementations of BPMN (74 listed)



Case studies

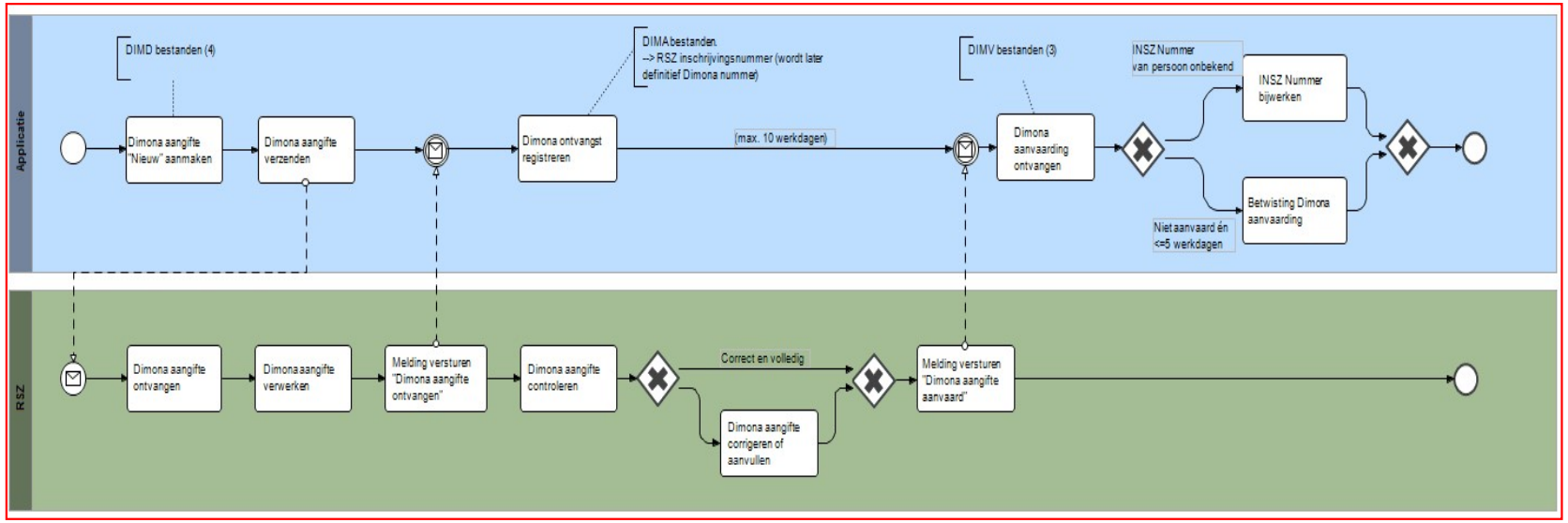
Project eProcurement / Vlaamse Overheid

Example: ‘ eProcurement Process ‘



Case studies

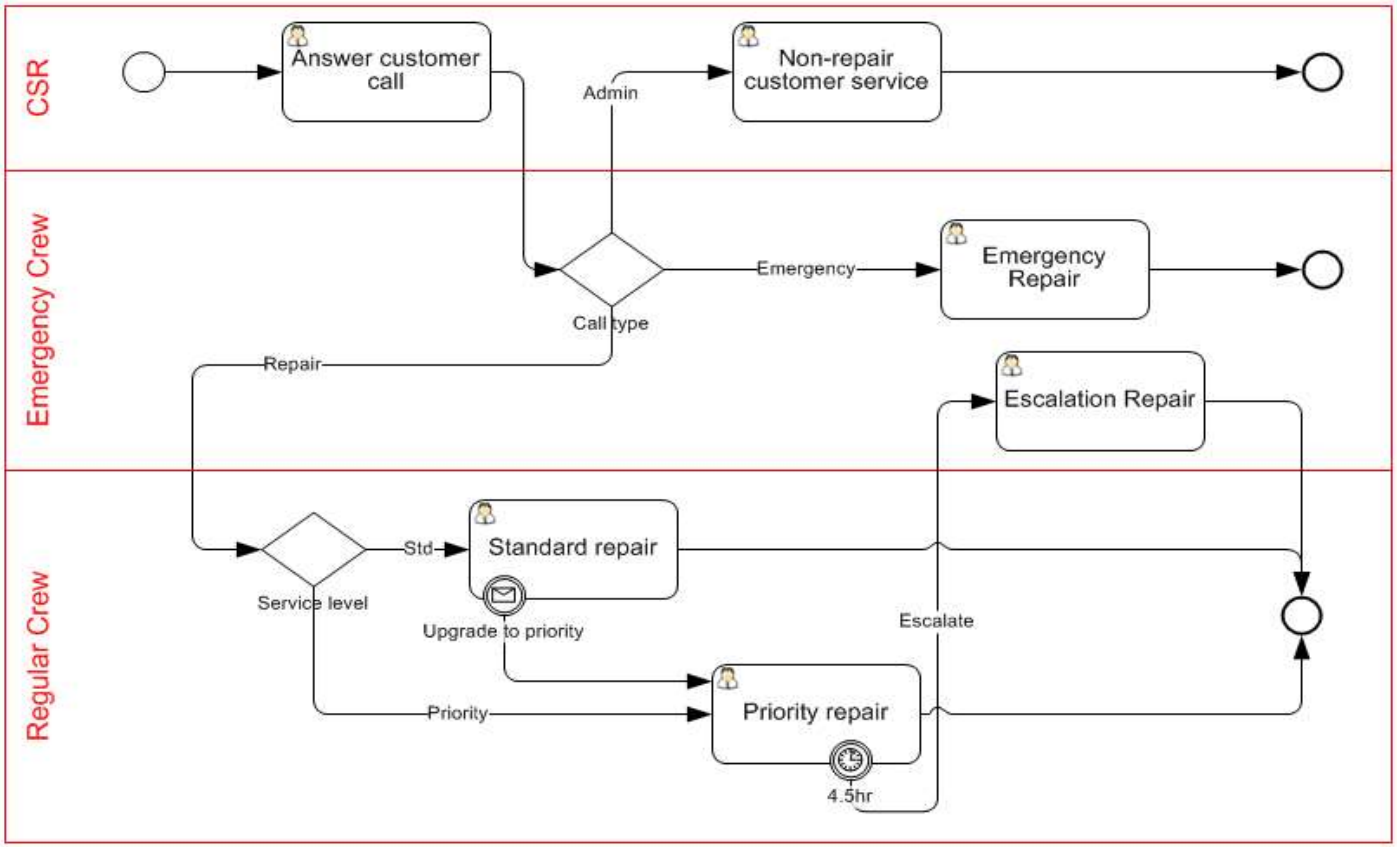
Project Dimona aangifte / Randstad



Case studies

Project Call Center / Company

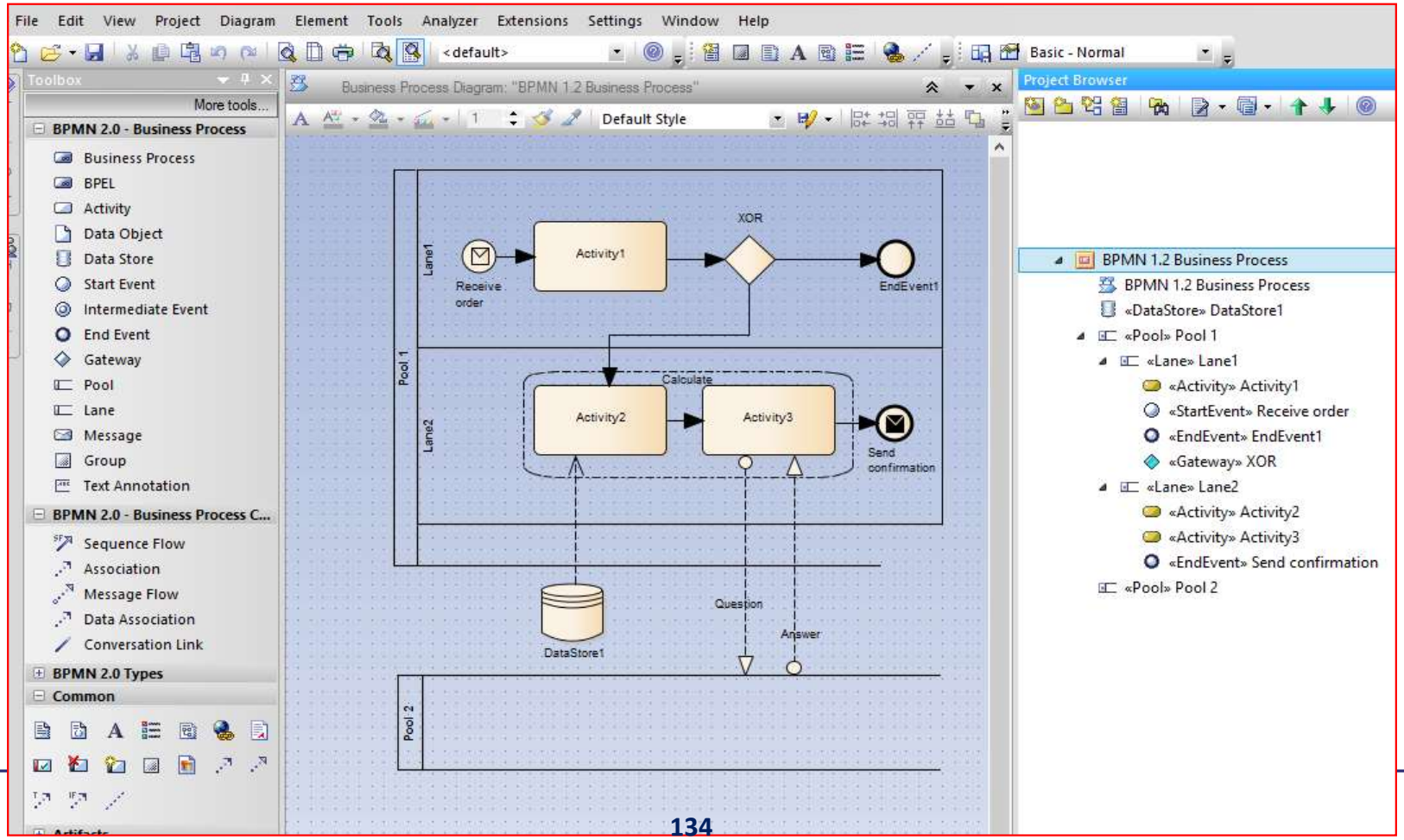
Process Modeler for Microsoft Visio™
Business Process Modeling



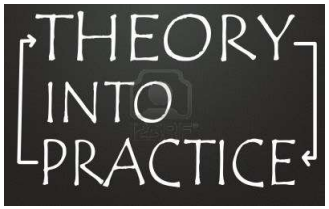
NCOI LEARNING

BPMN – Demo

Demo with Enterprise Architect



BPMN - Exercise

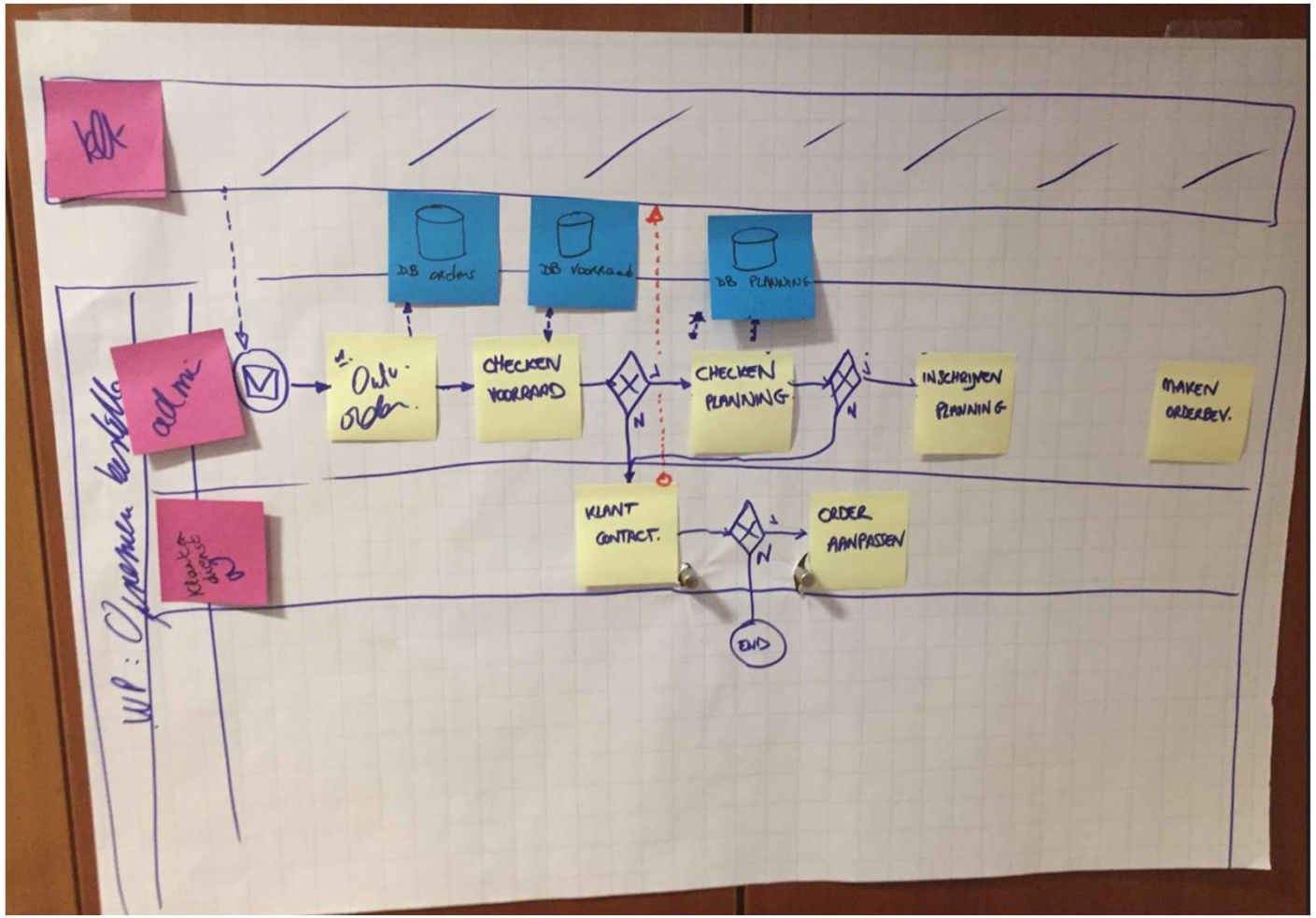


Exercise: Make a BPMN diagram

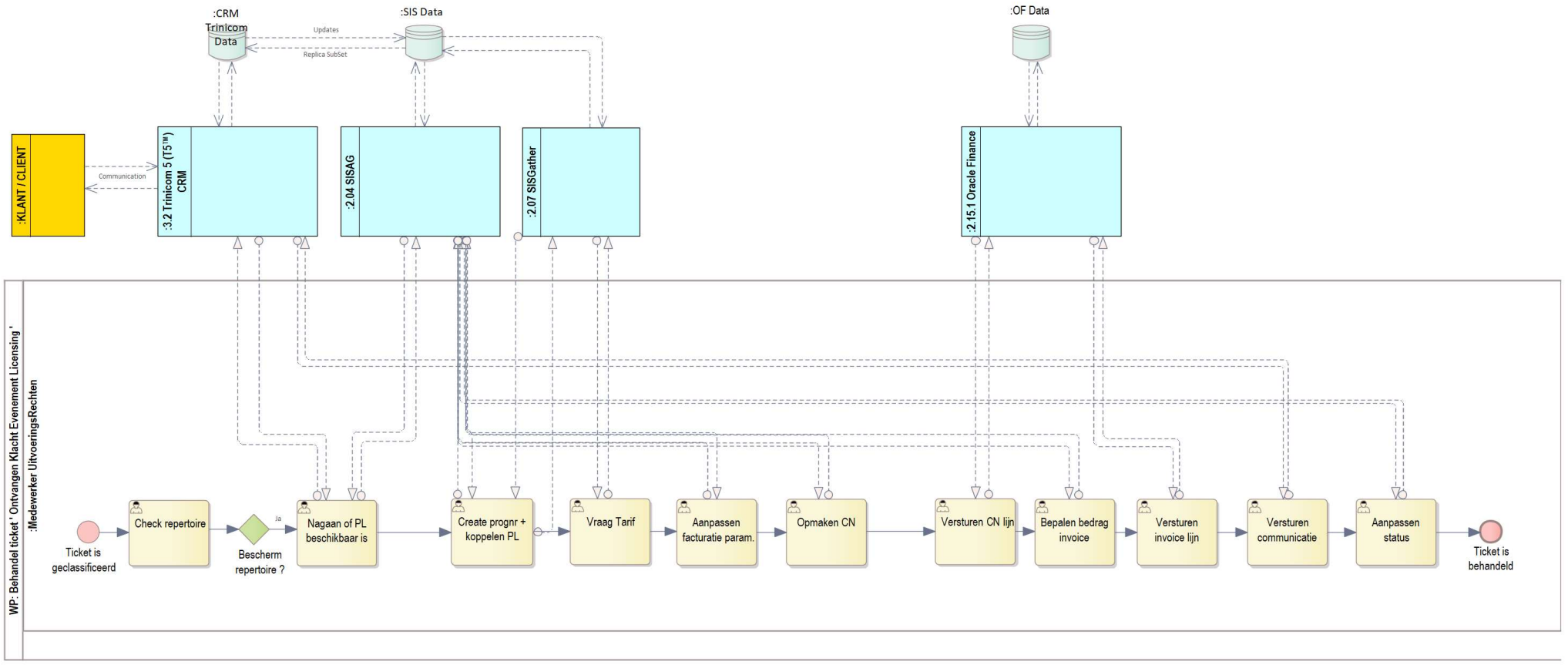
- Model the BPMN Diagram of the Brown Paper Session with your colleague where you both were involved in.

- Present your BPMN Diagram.

BPMN - Examples



BPMN - Examples



Part VIII: Functional models in UML

- Activity diagrams
- Sequence diagrams
- State diagrams
- Class diagrams
- Case studies
  - Real case examples
- Demo
- Exercise

SYNTAX OF UML

The UML diagrams

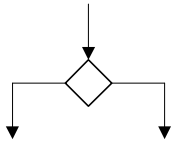
Activity diagrams

Basic symbols:

Initial node: a filled in circle which describes the starting point of the diagram.

Final node: a filled in circle with a border which describes the ending point.

Activity: a rounded rectangle that represents an activity. An activities can be physical (e.g. leave house) or electronic (e.g. transfer data)



Decision: a diamond with one flow entering and several leaving.

[Condition]

Condition guard: Text such as *[Incorrect Form]* on a flow, defining a guard which must evaluate to true in order to traverse the node.

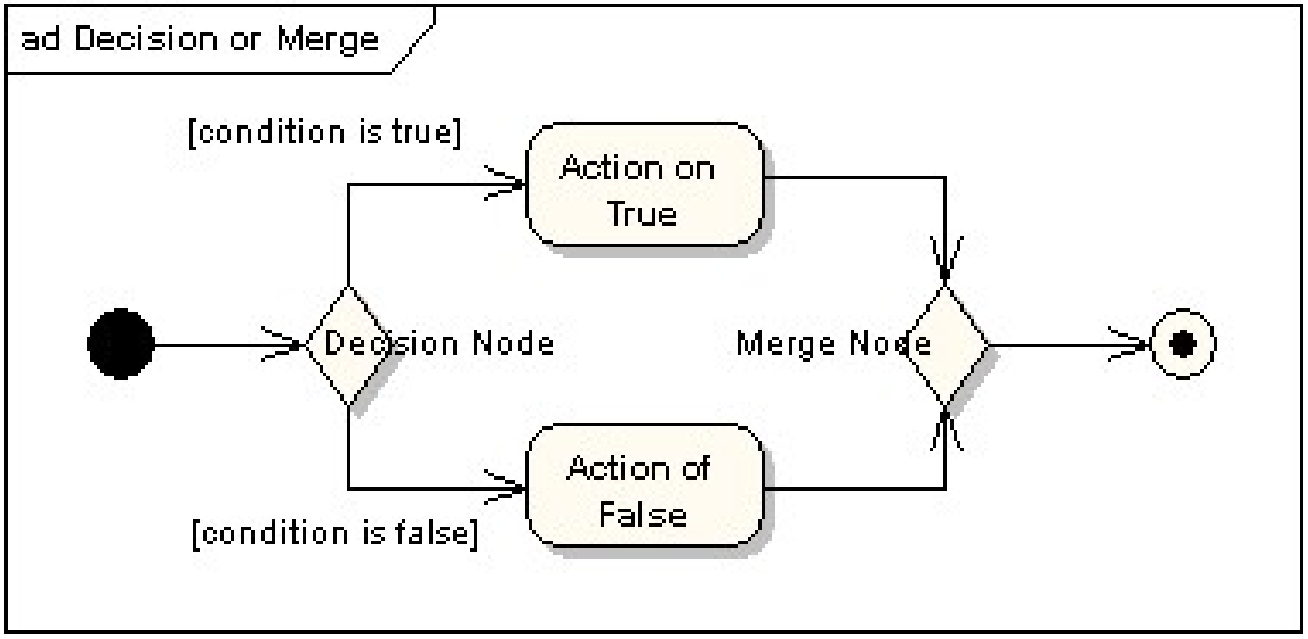
Flow: an arrow that connects activities, decisions an nodes.

SYNTAX OF UML

The UML diagrams

Activity diagrams

Example:

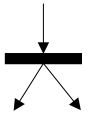


SYNTAX OF UML

The UML diagrams

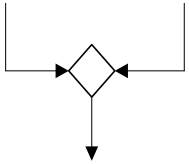
Activity diagrams

Symbols:

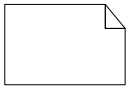


Fork: a node in an activity diagram where a flow is split into multiple parallel flows.

Join: a black bar with several flows entering it and one leaving it. All flows going into the join must reach it before processing may continue.



Merge: a diamond with several flows entering and one leaving. The implication is that one or more incoming flows must reach this point until processing continues, based on any guards on the outgoing flow.



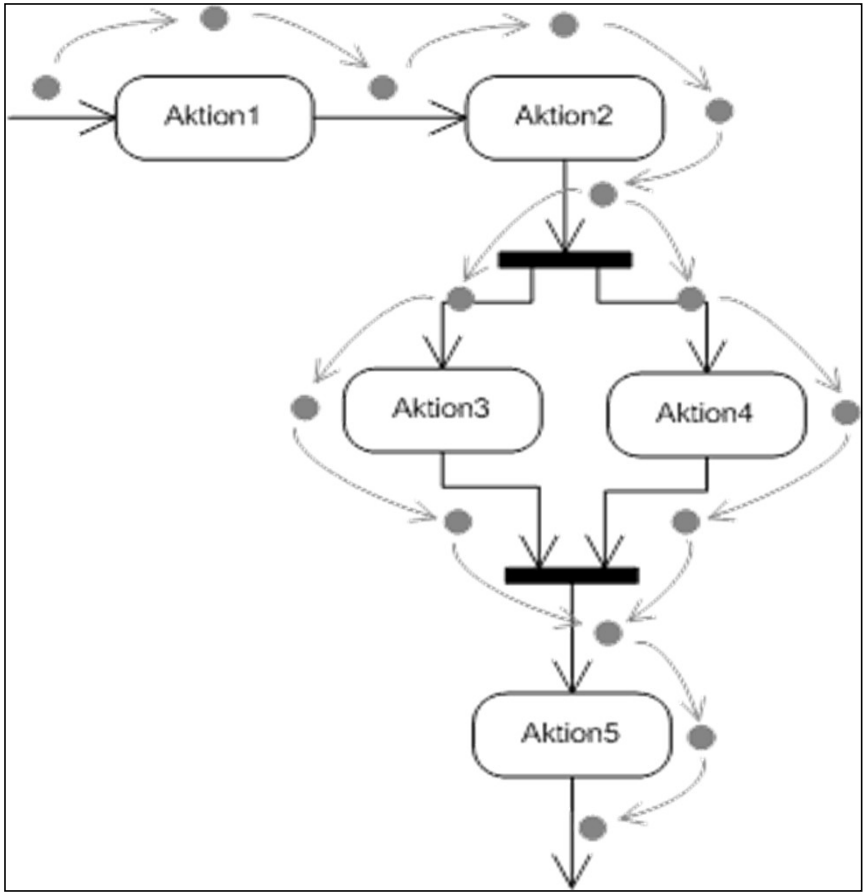
Note: adds some extra information to the diagram

SYNTAX OF UML

The UML diagrams

Activity diagrams

Example:

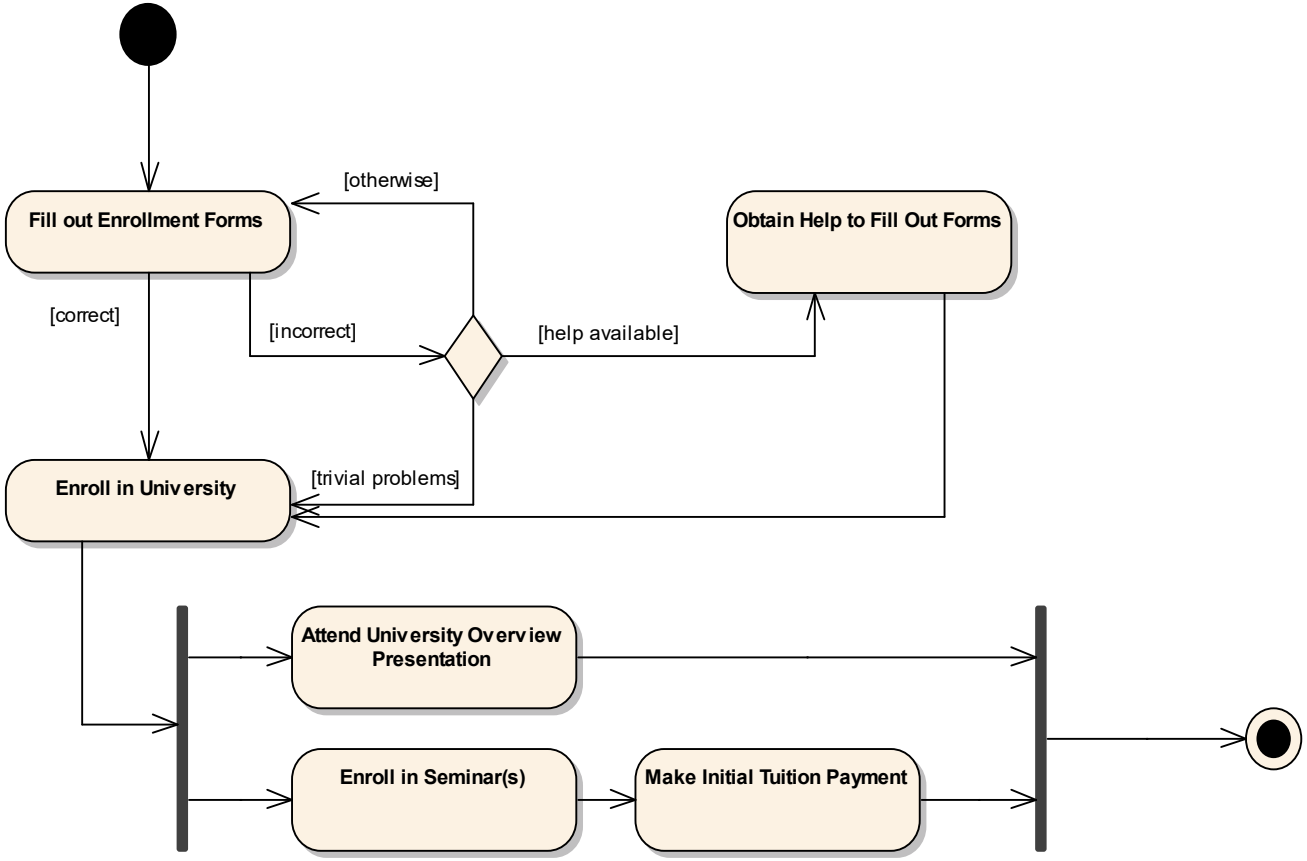


SYNTAX OF UML

The UML diagrams

Activity diagrams

Example:

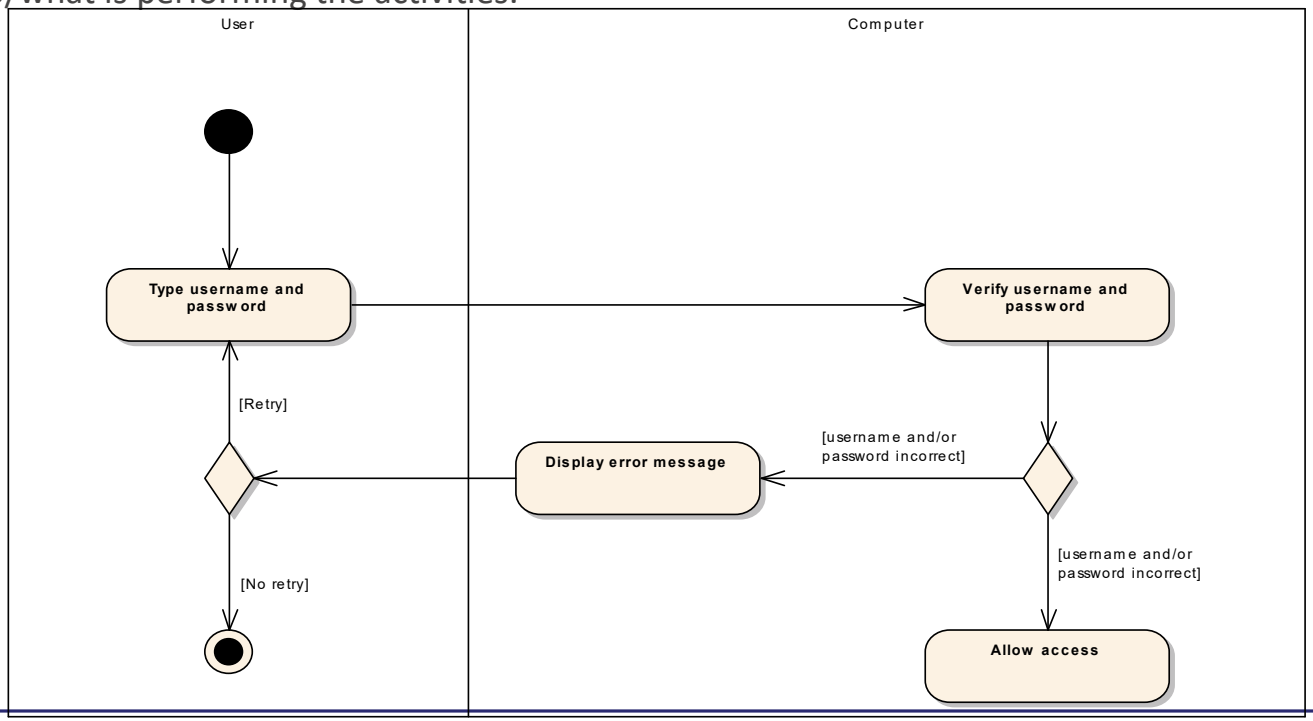


SYNTAX OF UML

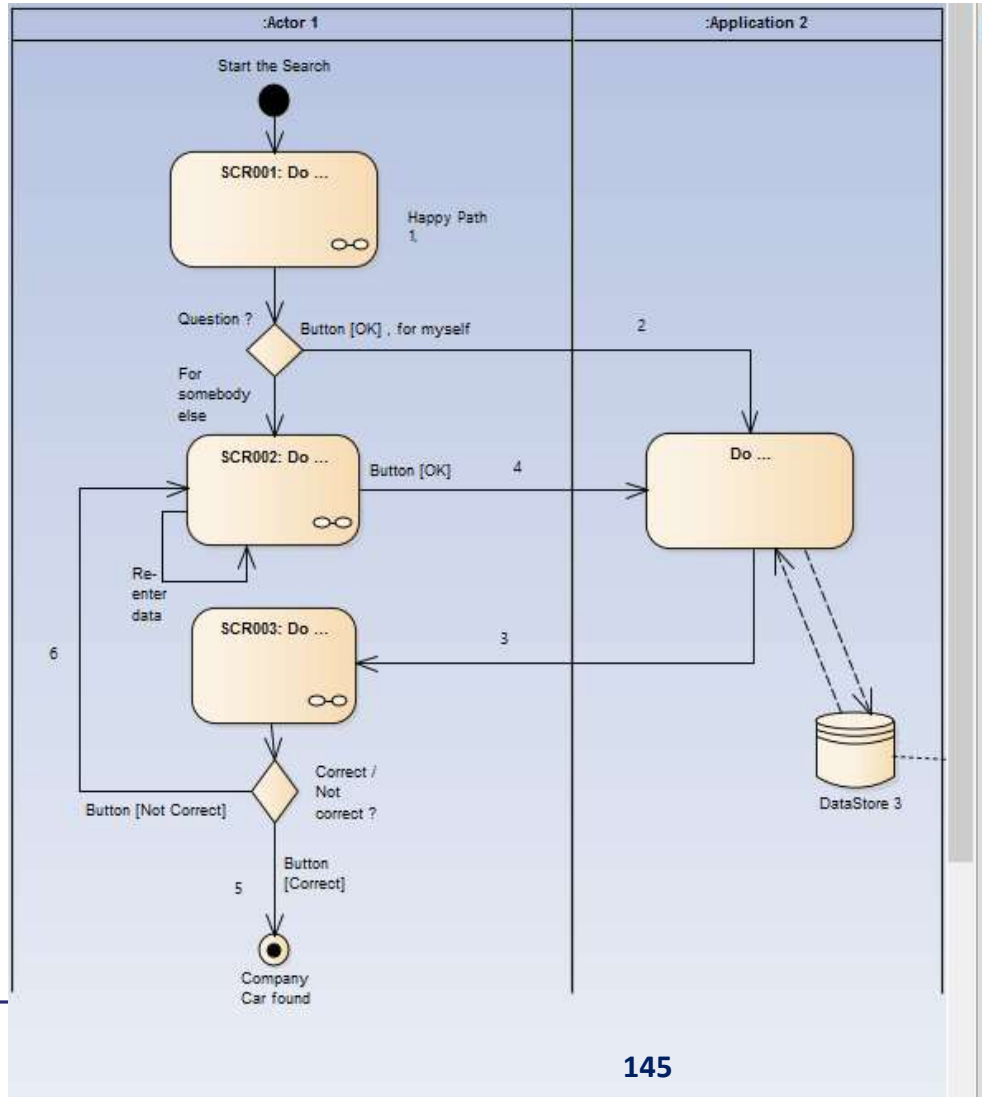
The UML diagrams

Activity diagrams

Partition or swimlane:
indicates who/what is performing the activities.



SYNTAX OF UML



SYNTAX OF UML

The UML diagrams

Activity diagrams

Best practices in drawing activity diagrams:

- Place the initial node in the top left corner.
- Always include a final node
- Avoid activities with no flow coming in or going out.

- A decision points always reflects the previous activity.
- Decision points don't have any label (unlike flow charts).
- Describe the decision in the out coming guards.

- Forks are the starting points of parallel activities. Joins are the ending points of parallel activities. So each fork should have a join.
- Swimlanes should be drawn vertically (though it's not obliged).

- Keep it simple: if you can't draw an activity diagram on one page, you need to reconsider your process.

Sources:
http://highered.mcgraw-hill.com/sites/0077110005/student_view0/glossary.html
http://www.agilemodeling.com/artifacts/activityDiagram.htm
http://www.agilemodeling.com/style/activityDiagram.htm

SYNTAX OF UML

The UML diagrams

Sequence diagrams

A second type of Interaction Diagrams are the Sequence Diagrams.
Sequence diagrams are sometimes also called event-trace diagrams, event scenarios or timing diagrams.

Sequence diagrams represent the interaction between objects and focus on

- the messages sent to other objects
- the messages received from other objects
- the order in which the messages occur

These messages are represented in a TIME dimension, as a kind of Message Sequence Chart.

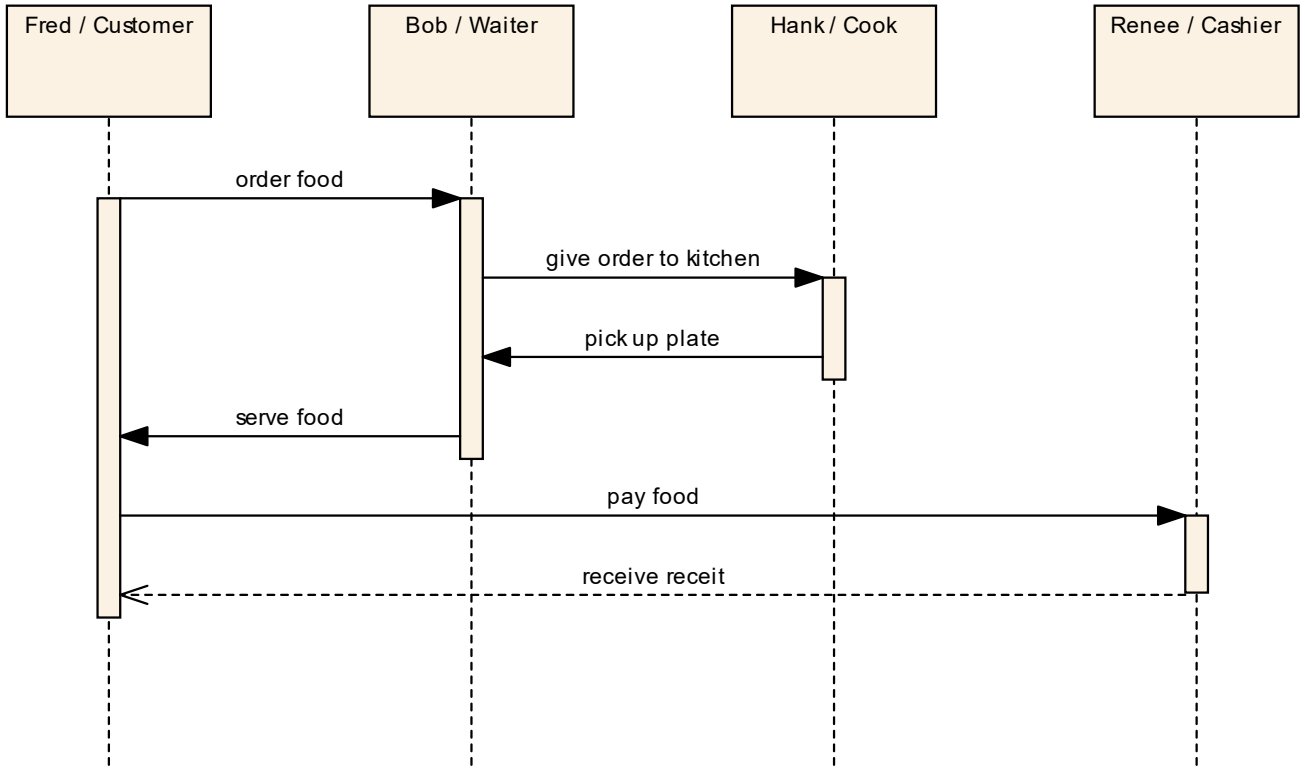
One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams

SYNTAX OF UML

The UML diagrams

Sequence diagrams

Example



SYNTAX OF UML

The UML diagrams

Sequence diagrams

Base Components within a Sequence Diagram:

Lifelines

- Lifelines represent either **roles or object instances** that participate in the sequence being modeled
- Lifelines are drawn as a box on **top** of the diagram with a dashed line descending from the center of the bottom edge. The lifeline's name is placed inside the box.
- When an **underline** is used, it means that the lifeline represents a **specific instance** of a class in a sequence diagram and not a particular kind of instance (i.e. a role).
- The **order** of the objects is **not relevant**, but to improve readability of the diagram, it is recommended to display the objects from left to right in order of appearance in the process.

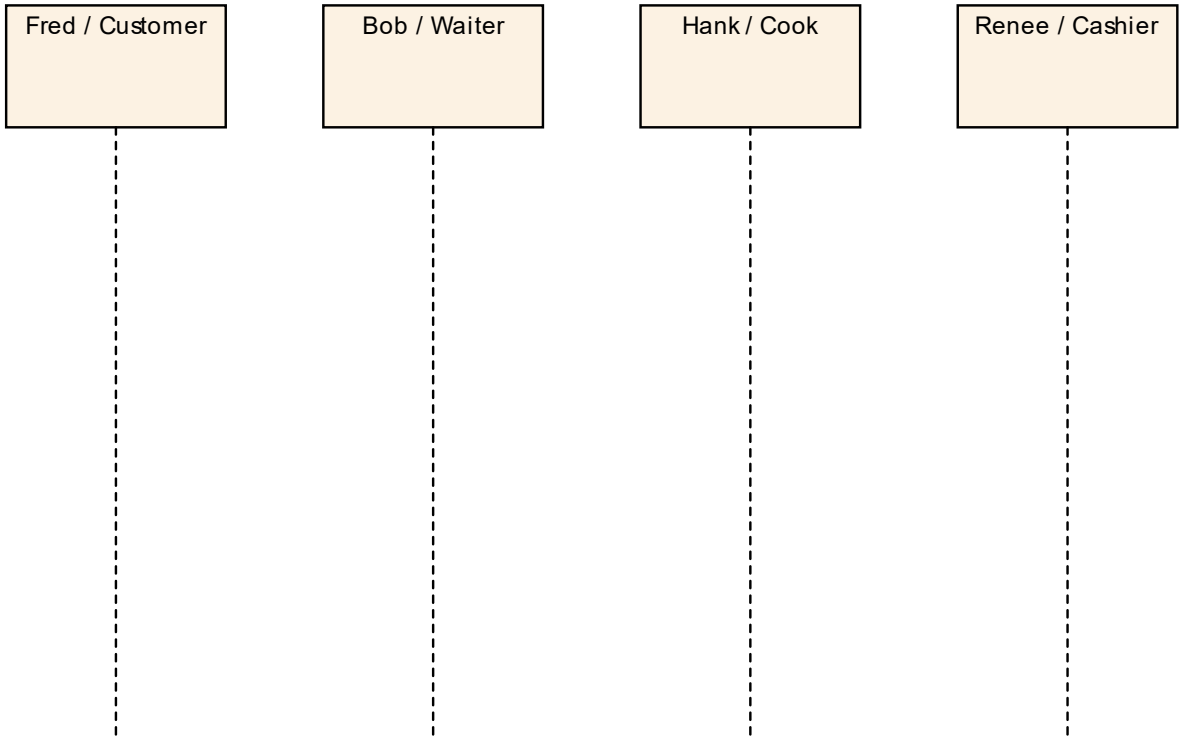
SYNTAX OF UML

The UML diagrams

Sequence diagrams

Base Components:

Lifelines



SYNTAX OF UML

The UML diagrams

Sequence diagrams

Base Components:

Messages

- The first message of a sequence diagram always starts at the top and is typically located on the left side of the diagram for readability.
- Subsequent messages are then added to the diagram slightly lower than the previous message.
- They are represented as horizontal arrows with the message name written above them:

  - Solid arrows with full heads are synchronous calls 
  - Solid arrows with stick heads are asynchronous calls 
  - Dashed arrows with stick heads are return messages. 

SYNTAX OF UML

The UML diagrams

Sequence diagrams

Base Components:

Return Messages

- These are optional. A return message is drawn as a dotted line with an open arrowhead back to the originating lifeline and above this dotted line the return value from the operation is placed.

- The use of return messages depends on the level of detail/abstraction that is being modeled. Return messages are useful if finer detail is required; otherwise, the invocation message is sufficient

SYNTAX OF UML

The UML diagrams

Sequence diagrams

Base Components:

Activation box

- Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message.

- They indicate the time during which the object instance is active.

- Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing.

SYNTAX OF UML

The UML diagrams

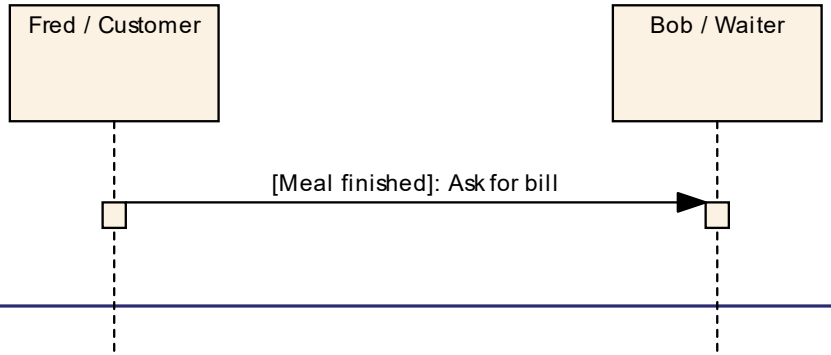
Sequence diagrams

Base Components:

Guards

- When modeling object interactions, there will be times when a condition must be met for a message to be sent to the object. Guards are used throughout UML diagrams to control flow.

- In UML 1.x, a guard could only be assigned to a single message. To draw a guard on a sequence diagram in UML 1.x, the guard element is placed above the message line being guarded and in front of the message name.



SYNTAX OF UML

The UML diagrams

Sequence diagrams

Combined fragments (alternatives, options, and loops) :

- In most sequence diagrams, however, the UML 1.x "in-line" guard is not sufficient to handle the logic required for a sequence being modeled.
- UML 2 has addressed this problem by removing the "in-line" guard and adding a notation element called a Combined Fragment.
- A combined fragment is used to group sets of messages together to show conditional flow in a sequence diagram.
- The UML 2 specification identifies 11 interaction types for combined fragments. Three of the eleven will be covered here.

SYNTAX OF UML

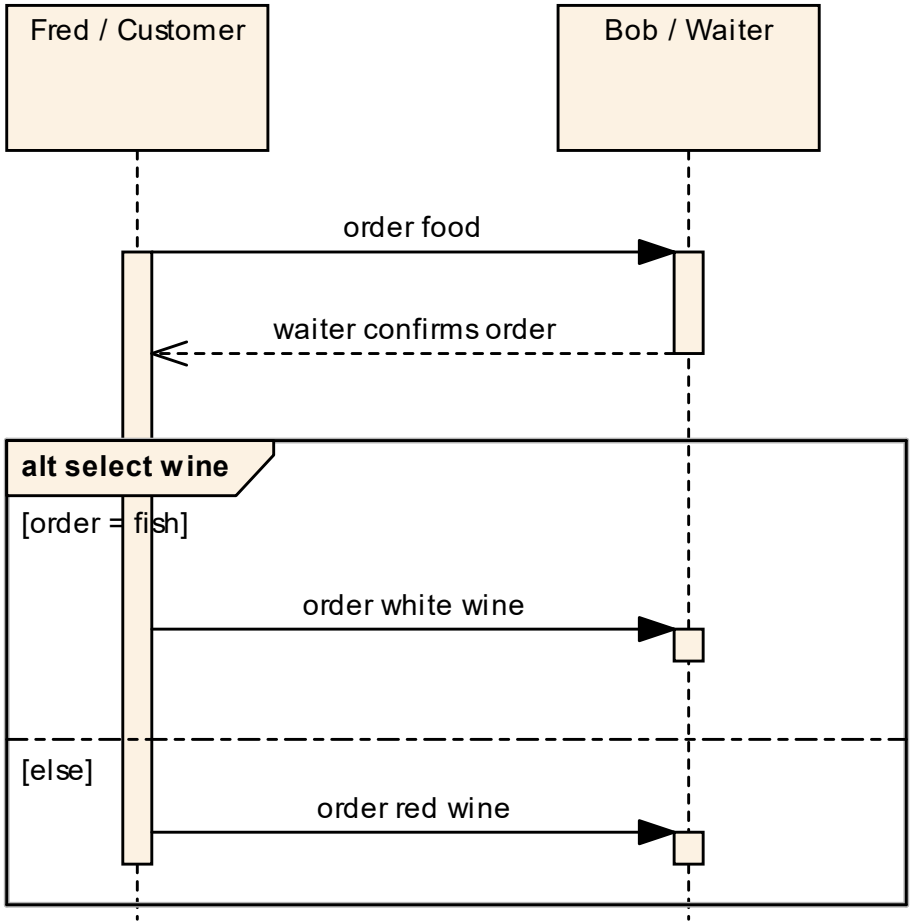
The UML diagrams

Sequence diagrams

Combined fragments:

Alternatives

- Alternatives are used to designate a mutually exclusive choice between two or more message sequences.
- Alternatives allow the modeling of the classic "if then else" logic .



SYNTAX OF UML

The UML diagrams

Sequence diagrams

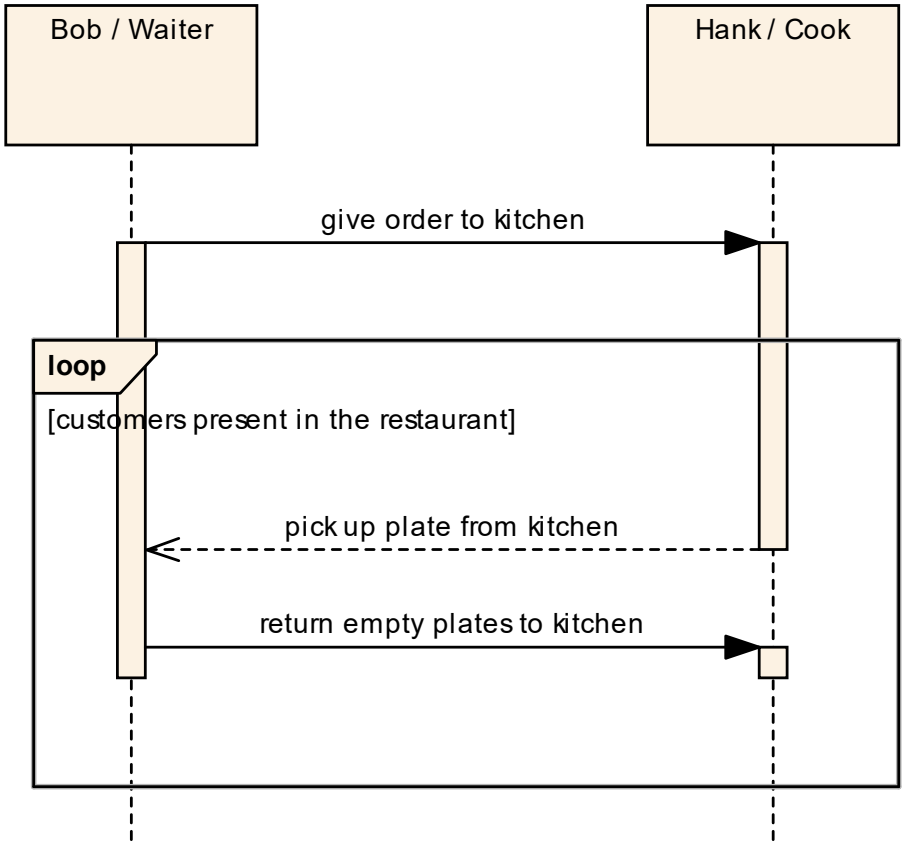
Combined fragments:

Loop

- Occasionally you will need to model a repetitive sequence.
- In UML 2, modeling a repeating sequence has been improved with the addition of the loop combination fragment.

Option

- The option combination fragment is used to model a sequence that, given a certain condition, will occur; otherwise, the sequence does not occur.
- An option is used to model a simple "if then" statement.



SYNTAX OF UML

The UML diagrams

Sequence diagrams

Synchronous communication:

- The object sending the message will wait for the process to terminate, or the reception of a return message.

Asynchronous communication:

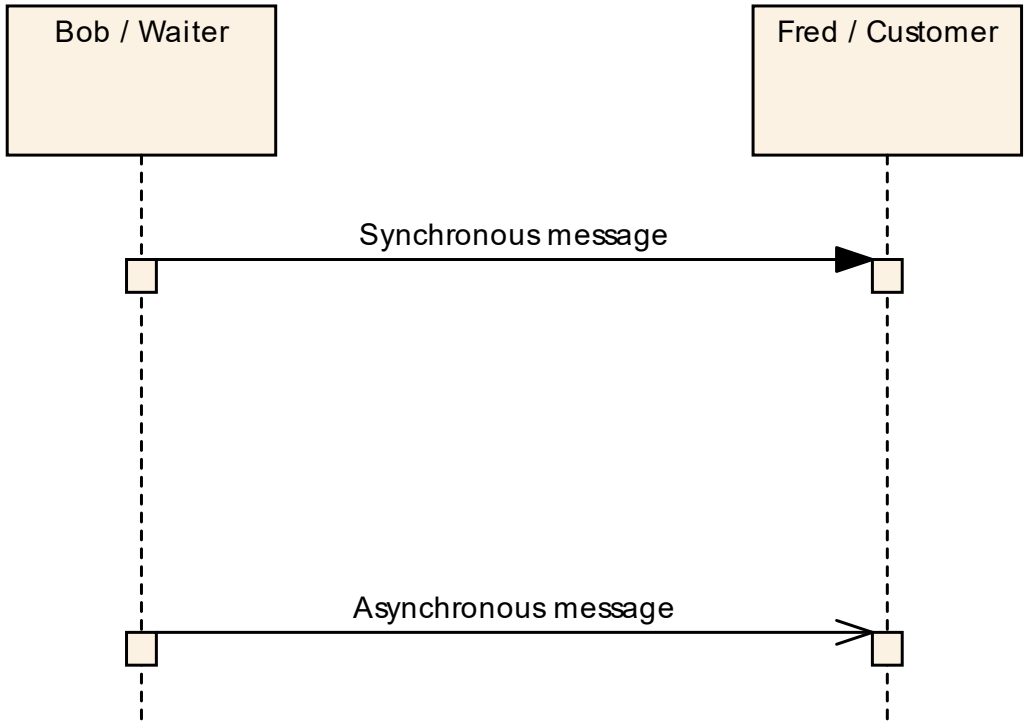
- The object sending the message will continue the process, without waiting for the process to terminate, or the reception of a return message.
- An asynchronous message is drawn similar to a synchronous one, but the message's line is drawn with a stick arrowhead.
- Asynchronous communication is mainly used in real-time systems where multiple processes can run simultaneously.

.

SYNTAX OF UML

The UML diagrams

Sequence diagrams

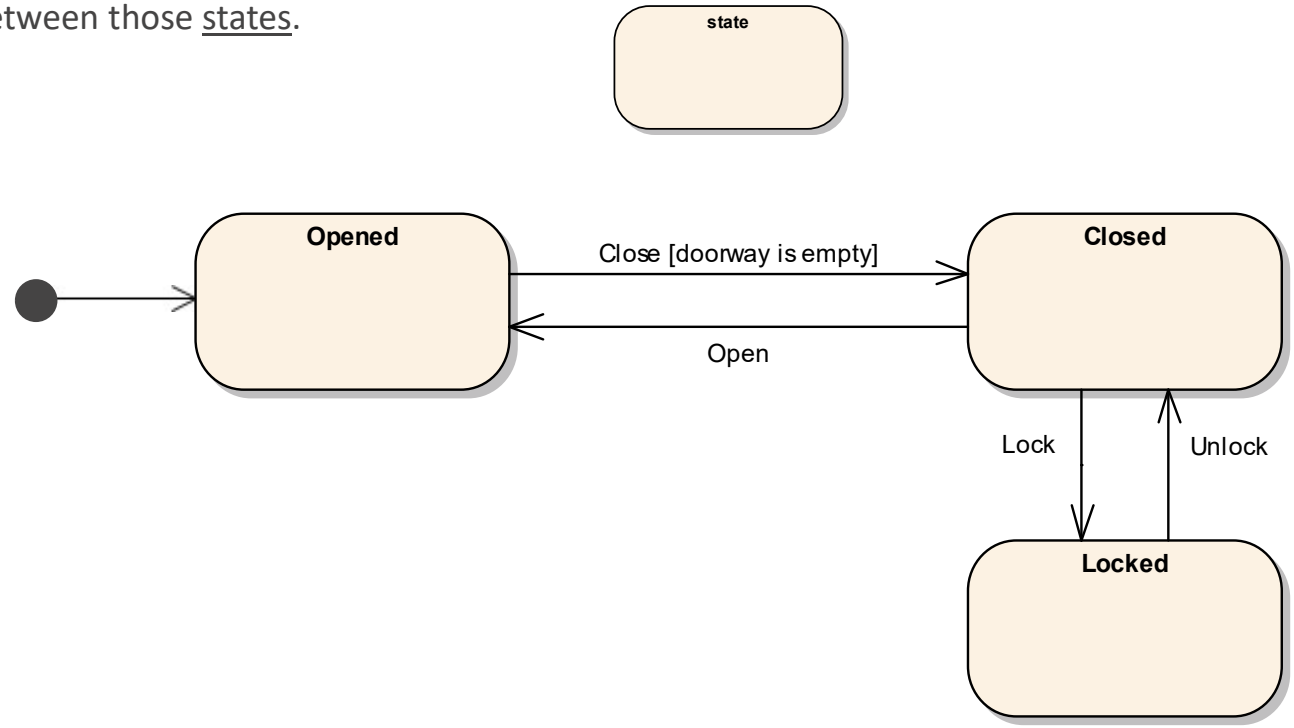


SYNTAX OF UML

The UML diagrams

State machine diagrams

A State Machine Diagram depicts the various states that an object may be in end the transitions between those states.

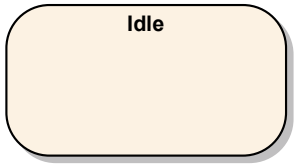


SYNTAX OF UML

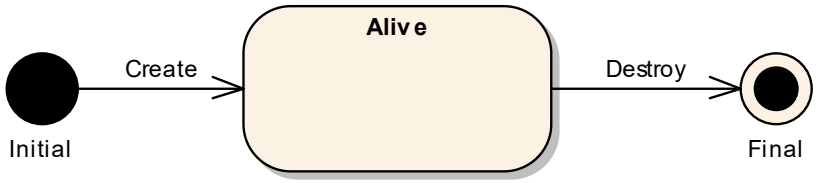
The UML diagrams

State machine diagrams

A state represents a stage in the behavior of an object. A state is denoted by a round-cornered box with the name of the state written inside it.



It is also possible to have initial states and final states. An initial state, also called a creation state, is the one that an object is in when it is first created. The initial state is denoted by a filled black circle and may be labeled with a name. A final state is a state in which no transitions lead out of, it is denoted by a circle with a dot inside and may also be labeled with a name.

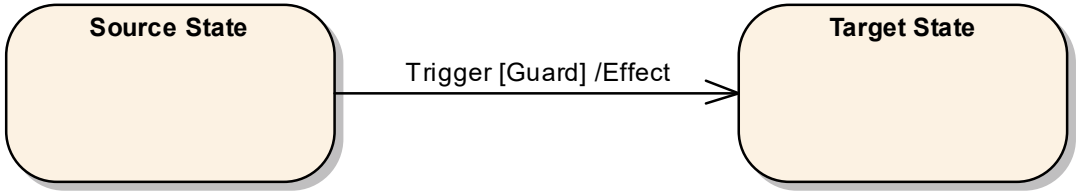


SYNTAX OF UML

The UML diagrams

State machine diagrams

A transition is a progression from one state to another and will be triggered by an event that is either internal or external to the object. Transitions from one state to the next are denoted by lines with arrowheads.



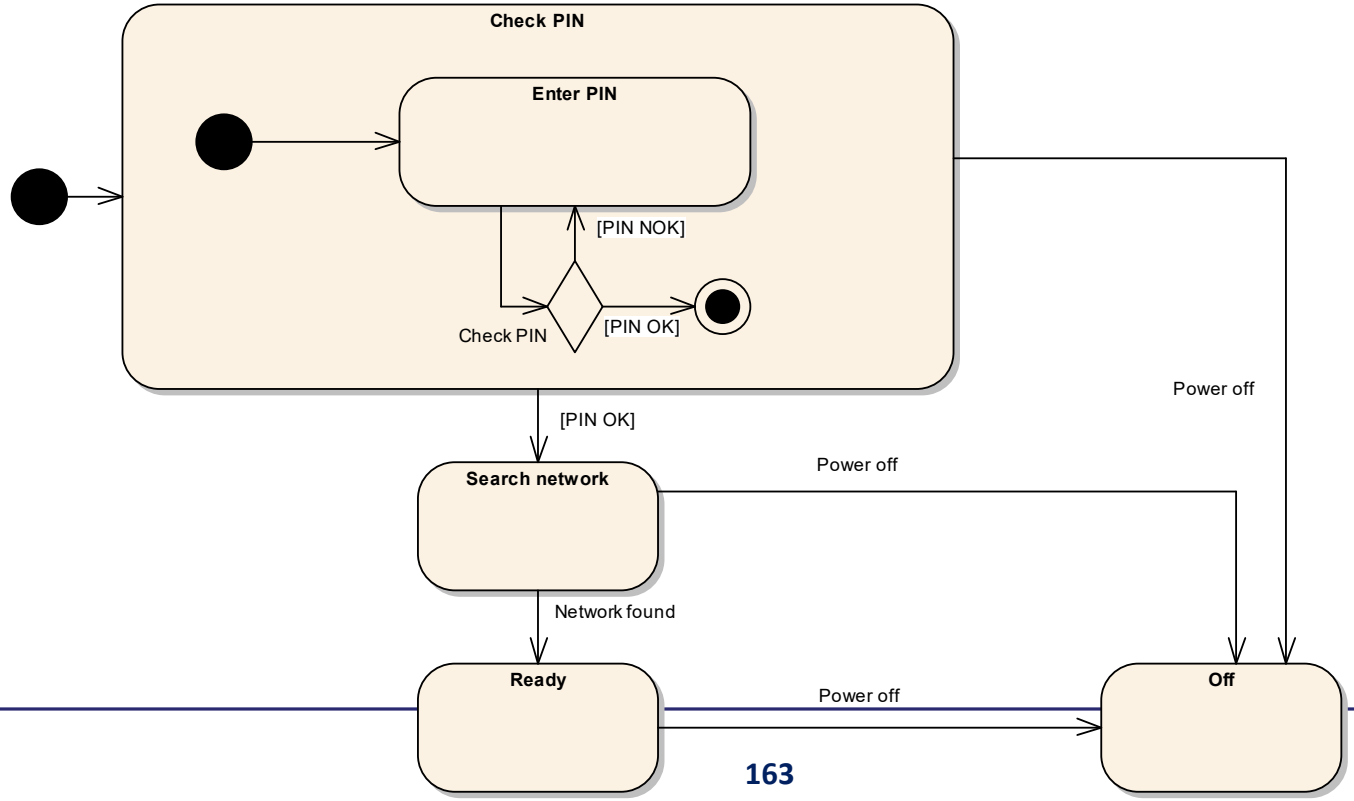
A transition may have a trigger, a guard and an effect. "Trigger" is the cause of the transition, which could be a signal, an event, a change in some condition, or the passage of time. "Guard" is a condition which must be true in order for the trigger to cause the transition. "Effect" is an action which will be invoked directly on the object that owns the state machine as a result of the transition.

SYNTAX OF UML

The UML diagrams

State machine diagrams

A state machine diagram may include sub-machine diagrams, called compound states, like this:

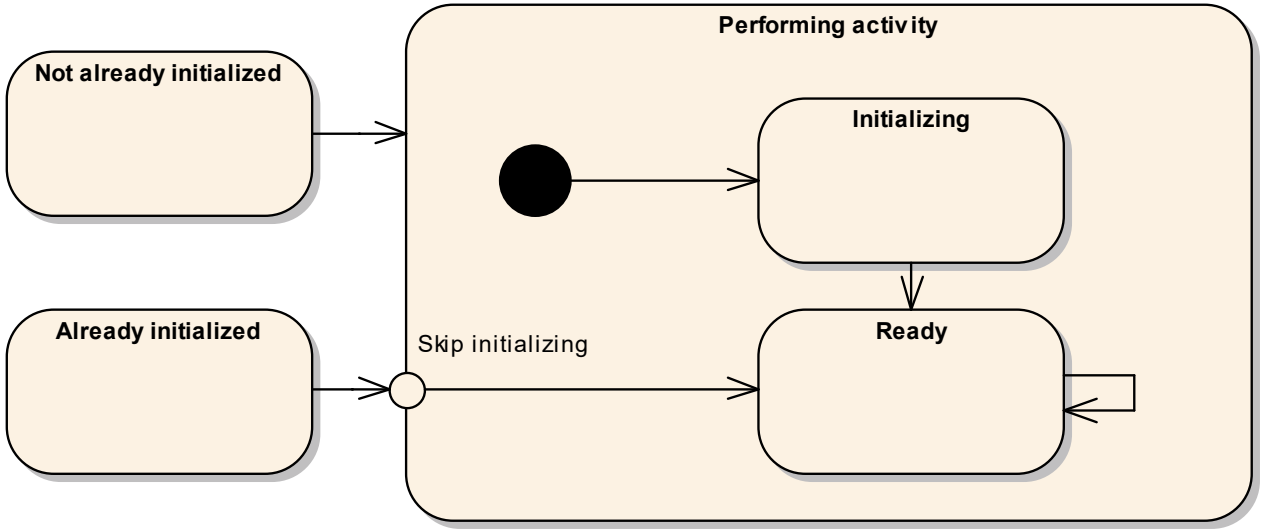


SYNTAX OF UML

The UML diagrams

State machine diagrams

Sometimes you won't want to enter a sub-machine at the normal initial state. For example, if for some reason it wasn't necessary to perform a certain transition, it would be possible to enter the sub-machine at a different named entry point.

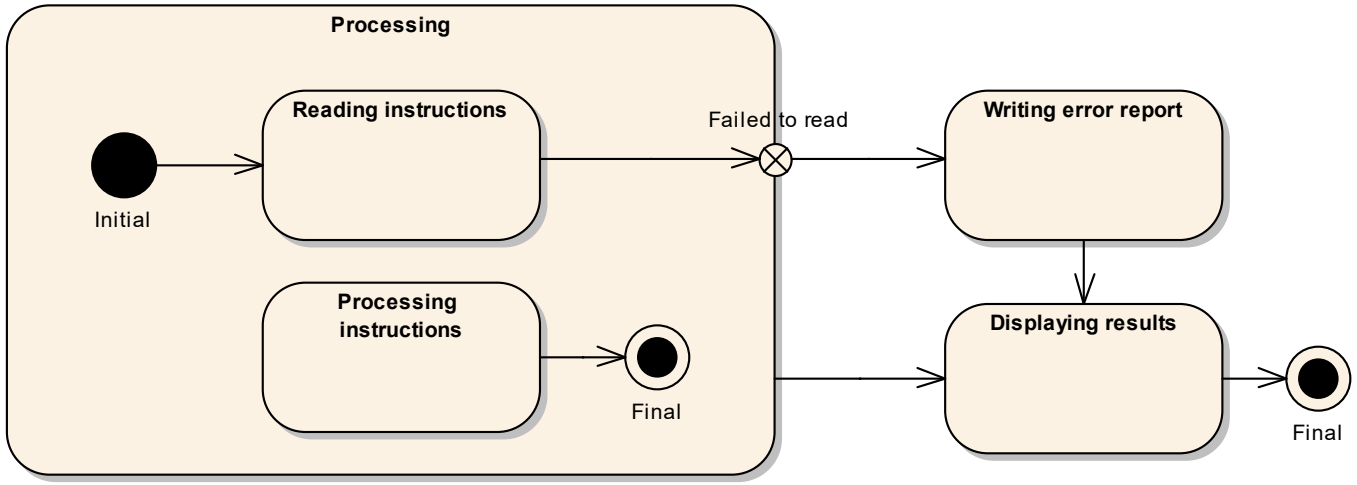


SYNTAX OF UML

The UML diagrams

State machine diagrams

In a similar manner to entry points, it is possible to have named alternative exit points.

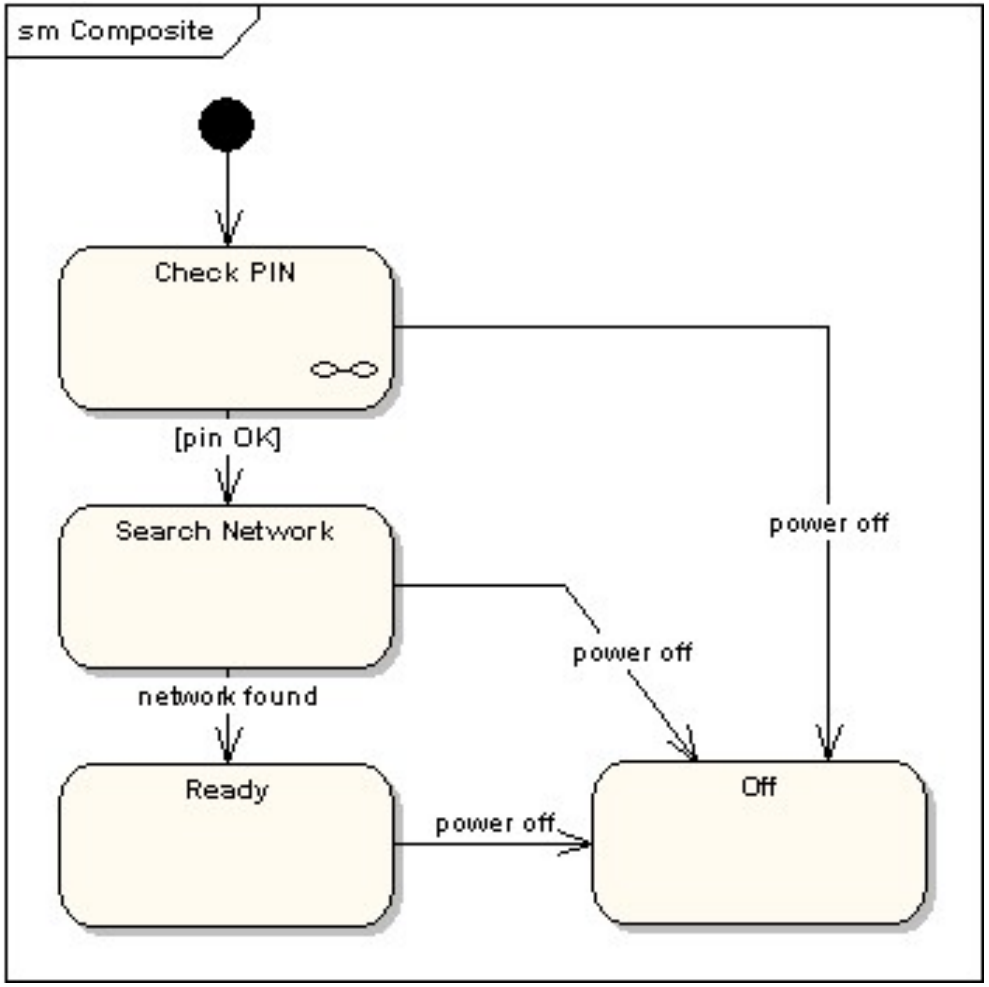


SYNTAX OF UML

The UML diagrams

State machine diagrams

Example:



SYNTAX OF UML

The UML diagrams

Class diagrams

Definition

A class diagram contains building blocks of an object oriented system and their relations.

A class is a collection of objects with according properties and operations.

A class diagram will grow during a project, and is build by modeling a user interface, identifying business logic, creating sequence diagrams, creating code,… .

SYNTAX OF UML

The UML diagrams

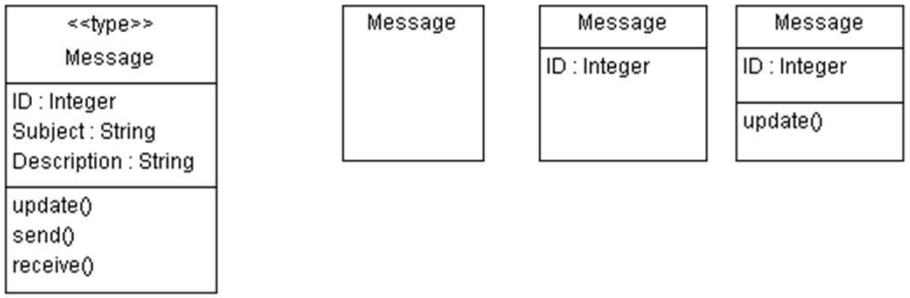
Class diagrams

Notation

A class is pictured with three compartments:

- Name and stereotype: the upper compartment contains the name, and optionally the stereotype of the class (example: stereotype="type", name="Message")
- Attributes: the middle compartment contains the attributes (example: "ID", "Subject" and "Description")
- Operations: the lowest compartment contains the operations of the class (example: "update()", "send()", "receive()")

The compartments are optional (only if not ambiguous) .



The image shows 4X the same class; there is no contradiction here.

UML is an additive specification method.

SYNTAX OF UML

The UML diagrams

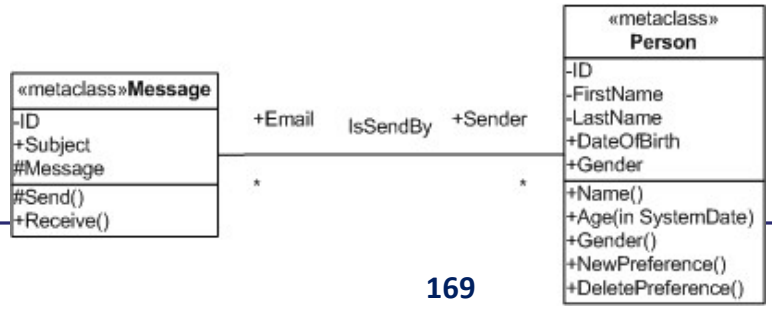
Class diagrams

Association

An association is a structured relationship between 1 or 2 classes. Structured means: an instance of class A has, during most of his existance, a connection with an instance of class B (A person will have, during most of his career, a relationship with one or more companies. This person won't be unemployed during most of the time.).

An association is represented by a single line, and has the following properties:

- Name (IsSendBy)
- Roles (the class(es): Message, Person)
- Reading direction (By default, relationships are red from left-right and up-down. In the example: Email-IsSendBy-Sender. Optional: arrow shows reading direction)
- Navigation (uni-directional, bi-directional, not specified): which class knows the other class of the association



SYNTAX OF UML

The UML diagrams

Class diagrams

- Multiplicity (*..*)
  - Notation: Minimum..Maximum
  - Minimum and maximum are numbers or '*' (=unlimited)

    0..* 1 instance of Class A has a relation with zero or more instances of Class B (Equivalent: '*')
    Zero to many

    1..* 1 instance of Class A has a relation with one or more instances of Class B (thus: relationship isn't optional)
    One to many

    .. 1 instance of Class A has a relation with one or more instances of Class B. And 1 instance of Class B has a relation with one or more instances of Class A
    Many to many.

  - Tip: add the multiplicity when it's different from '1'
- More than one association is allowed between two classes (between person and company: isCustomer and isEmployerOf).
- Reflexive associations: a class can have an association with herself (Association between Message and Message: 'Is send to').

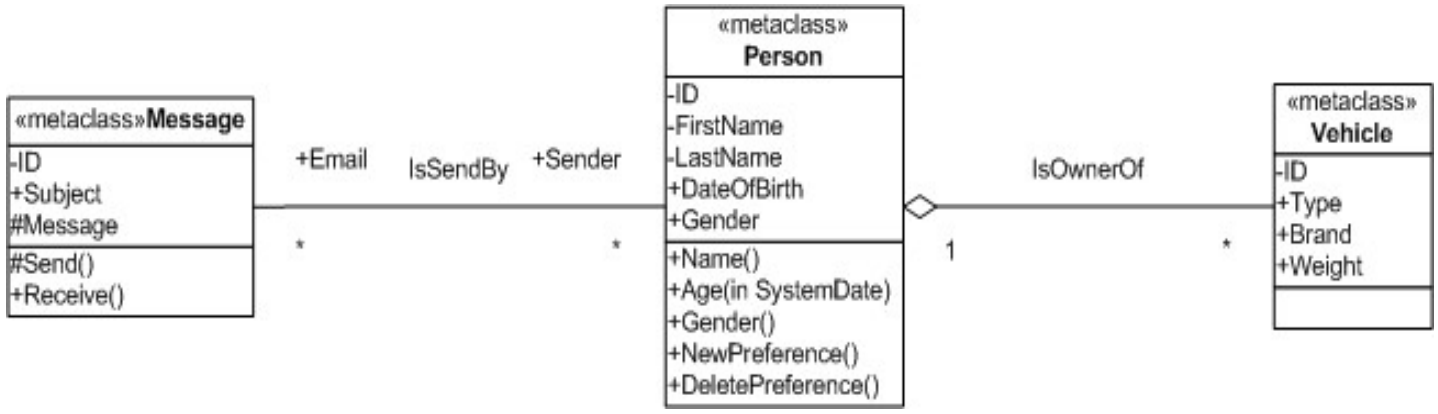
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced associations: Aggregation

- An aggregation is an association between an entity (Class A) and an element (Class B).
- An element can be used independently.
- An aggregation is heritable: if Class C is part of Class B, and Class B is part of Class A, then is Class C also a part of Class A.
- Representation: an association with a white diamond



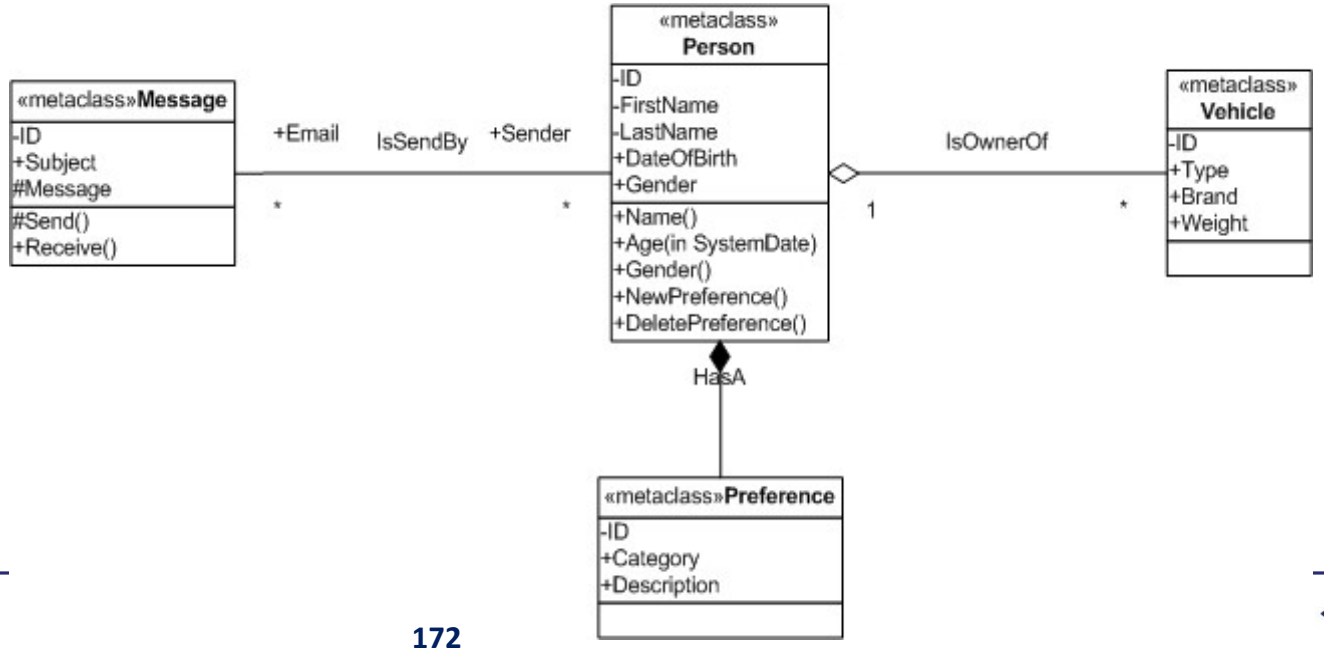
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced associations: Composition

- A composition is an association between an entity (Class A) and an element (Class B).
- An element is always a part of an entity (can't be used independently).
- Representation: an association with a black diamond



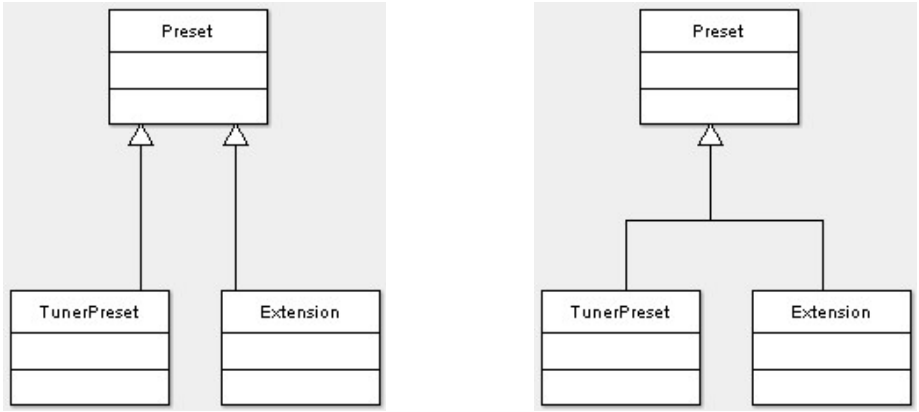
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Generalization and inheritance

- A generalization is a relation between a more general modeling element and a more specific modeling element (can be used for actors, use cases, classes).
- In a class diagram there is generalization between a superclass and a subclass.
- Visualization: line with a white triangle



- Both diagrams above have the same semantics.

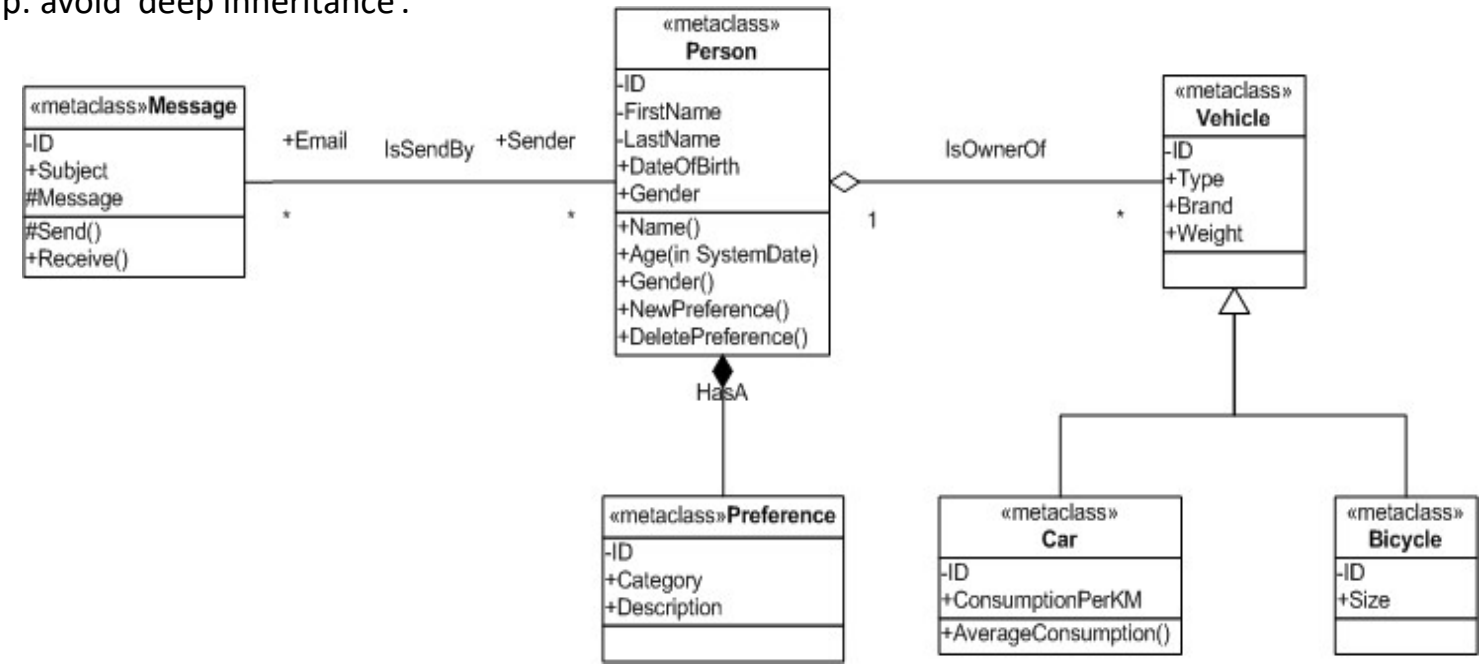
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Generalization and inheritance

- Tip: avoid ‘deep inheritance’.



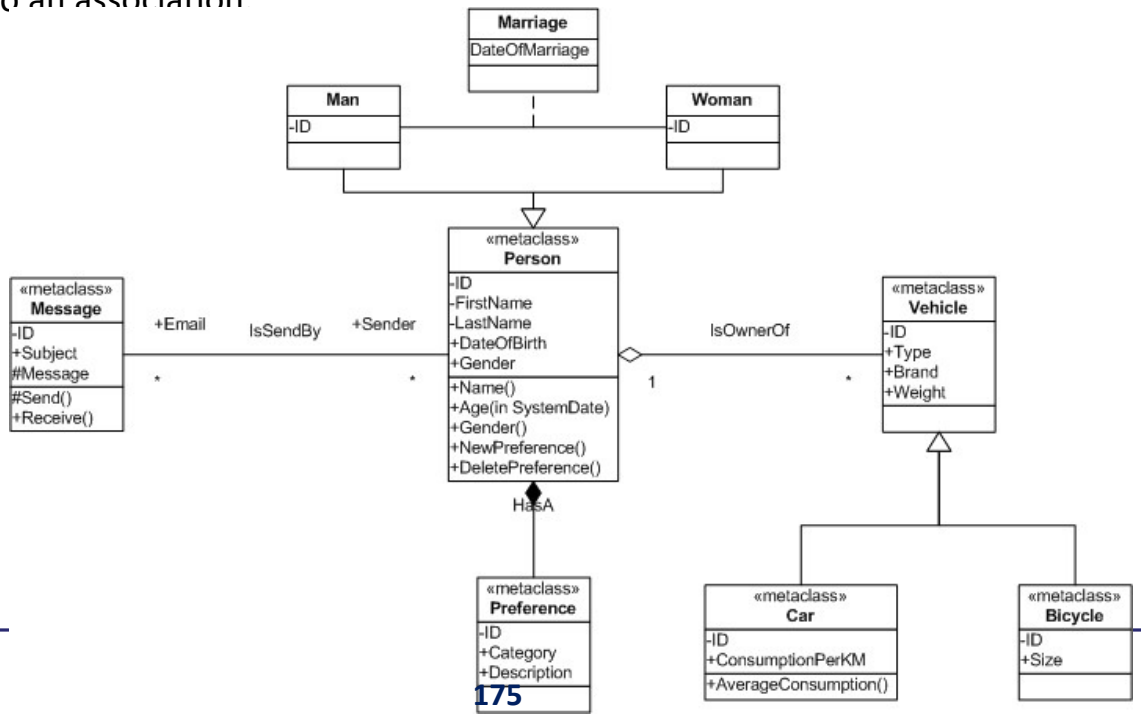
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Association class

- An association class is an association with the extra properties of a class attached to it. It is visualized by a class joined to an association



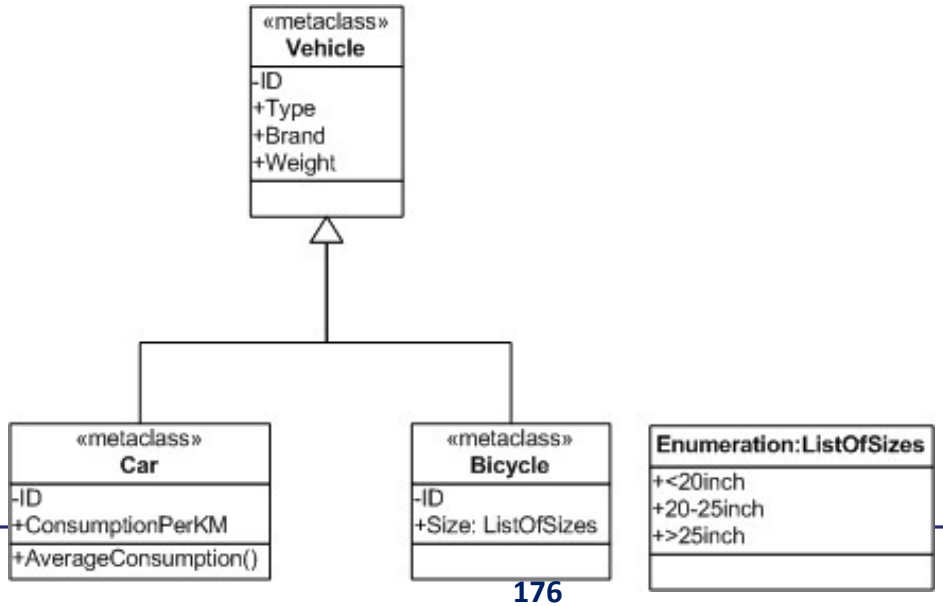
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Enumeration

- An enumeration is a collection of predefined constants called literals.
- When you want to assign a value to an attribute out of a short list of values, create an enumeration.
- Example: Attribute Size of class Bicycle will have a value of enumeration class ListOfSizes



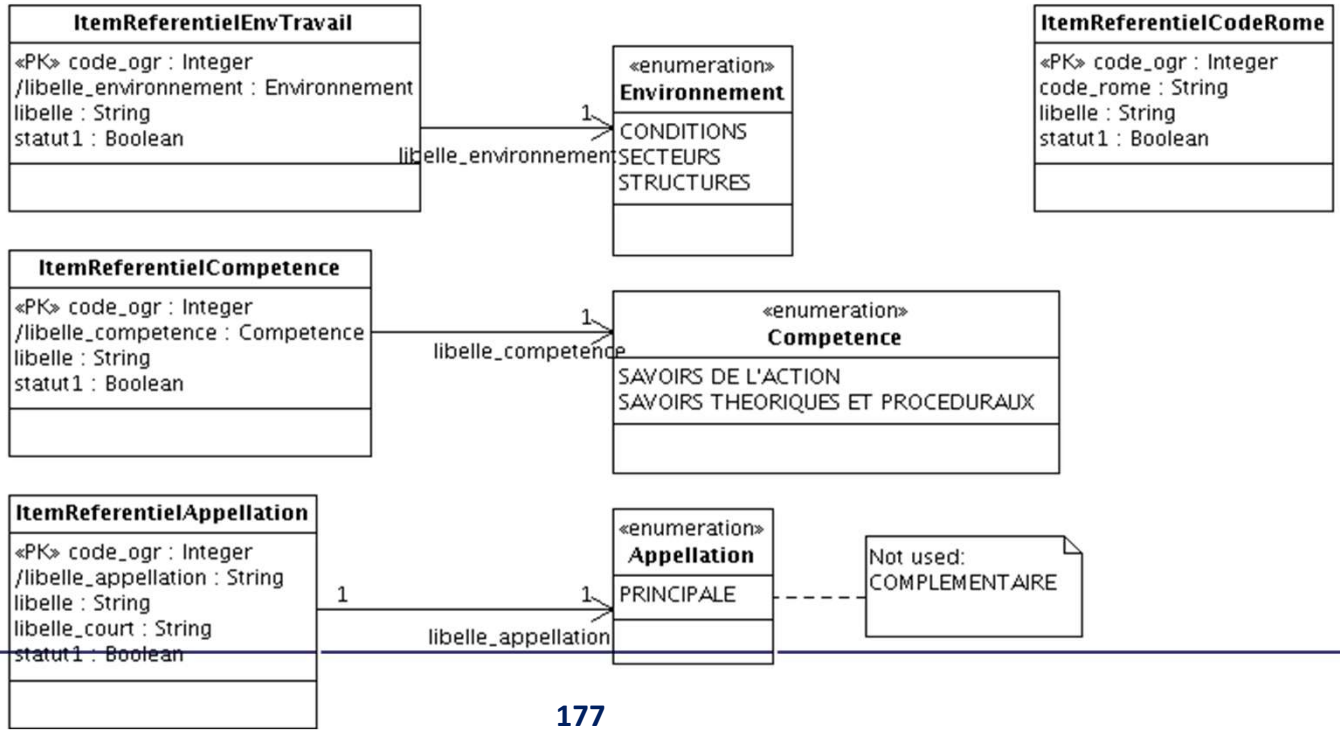
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Enumeration

Example of a data model that uses enumerations



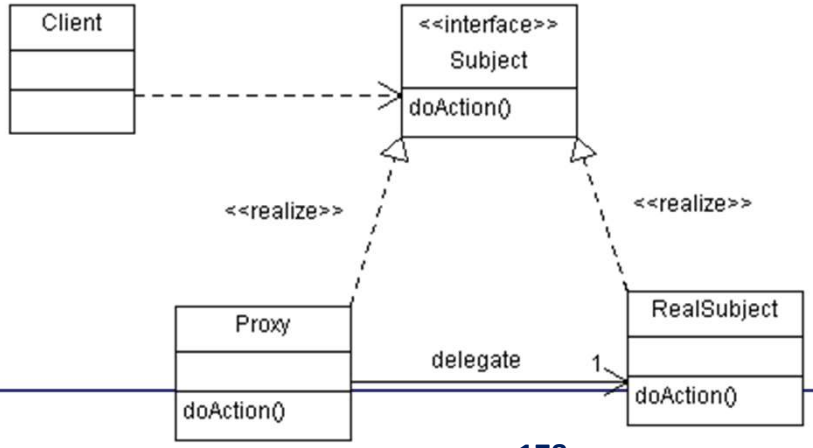
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Interfaces

- An interface is a contract for classes that realize the interface. It's a collection of external visible operations who describe the behavior of one or more classes.
- An interface is pictured by a class with operations but without attributes. The stereotype is "interface".
- Interfaces force polymorphism: very different classes will be approached in exact the same way. Frameworks often use interfaces (example: Microsoft .NET).
- Example: The "proxy" design pattern.



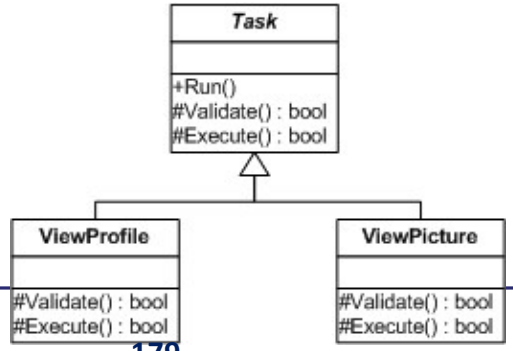
SYNTAX OF UML

The UML diagrams

Class diagrams

Advanced classes: Abstract classes

- An abstract class is a class without instances. It can contain normal operations, but has to have at least 1 abstract operation. Classes who inherit from the abstract class must implement the abstract operation. Thus: a concrete class has no abstract operations and an abstract class can have concrete operations. The subclasses differ to each other in the way they implement the abstract method(s).
- Difference with an interface:
  - If some classes must have similar behavior and part of the implementation is the same for all these classes: use an abstract class.
  - If some classes must have similar behavior and the implementation is unique per class: use an interface.

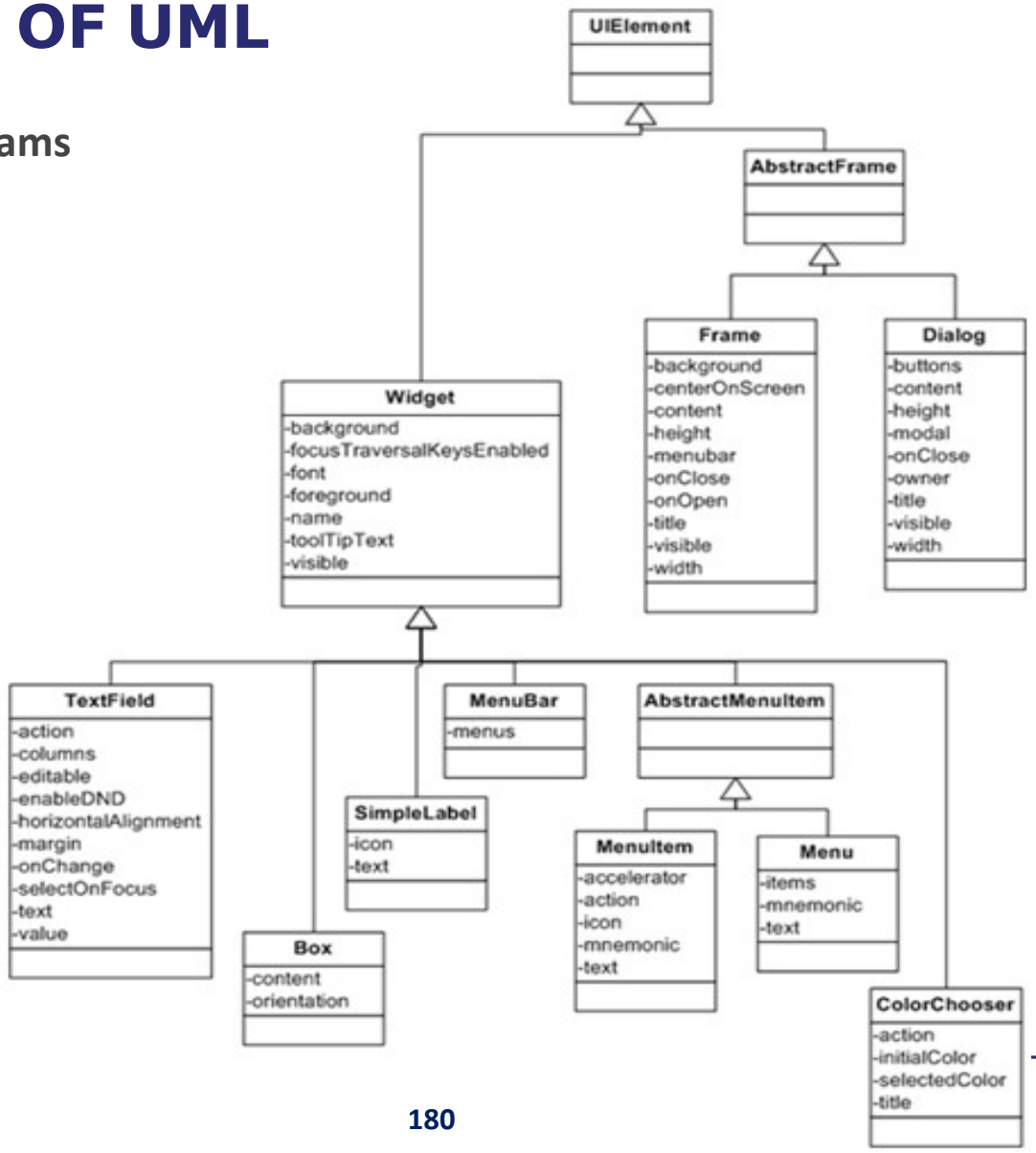


SYNTAX OF UML

The UML diagrams

Class diagrams

Example



SYNTAX OF UML

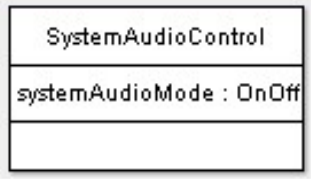
The UML diagrams

Class diagrams

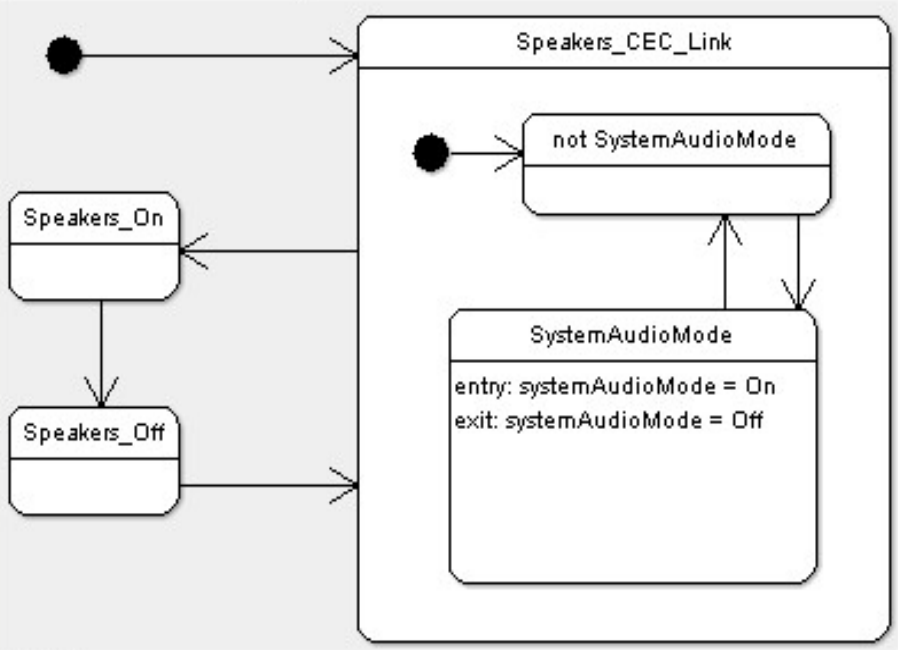
Example

This example demonstrates how to use an attribute from a class diagram on a state machine diagram.

12089



12090



Remark:
1. Triggers and guards are not shown in above diagram, but it does contain all possible transitions.
2. The 3 states which names start with "Speakers_", represent the Speakers menu-item position.

SYNTAX OF UML

The UML diagrams

Class diagrams

Example

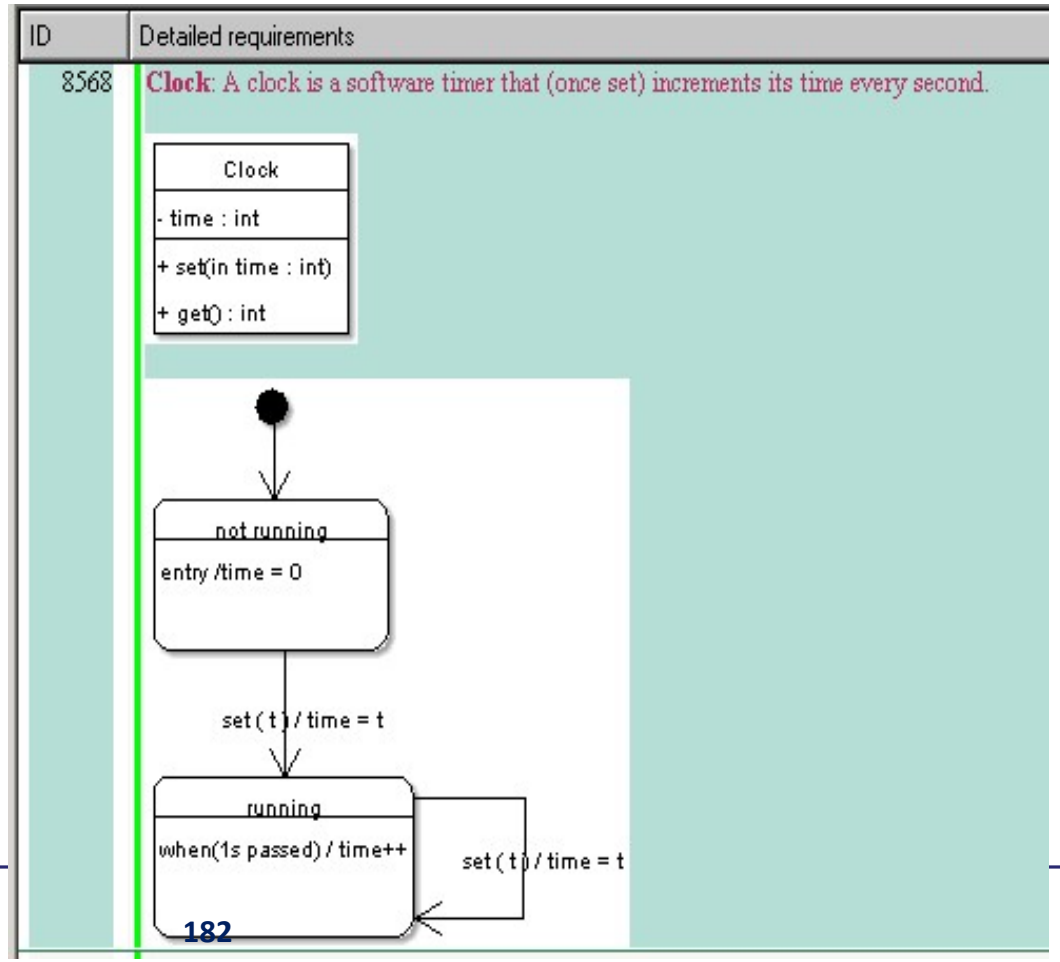
This example demonstrates how to use attributes and operations from a class diagram on a state machine diagram.
Also, it shows the Java code that certain tools can generate for the class diagram.

Generated Java source:
Clock.java

```
public class Clock {
  private int time;

  public void set(int time) {
  }

  public int get() {
    return 0;
  }
}
```



SYNTAX OF UML

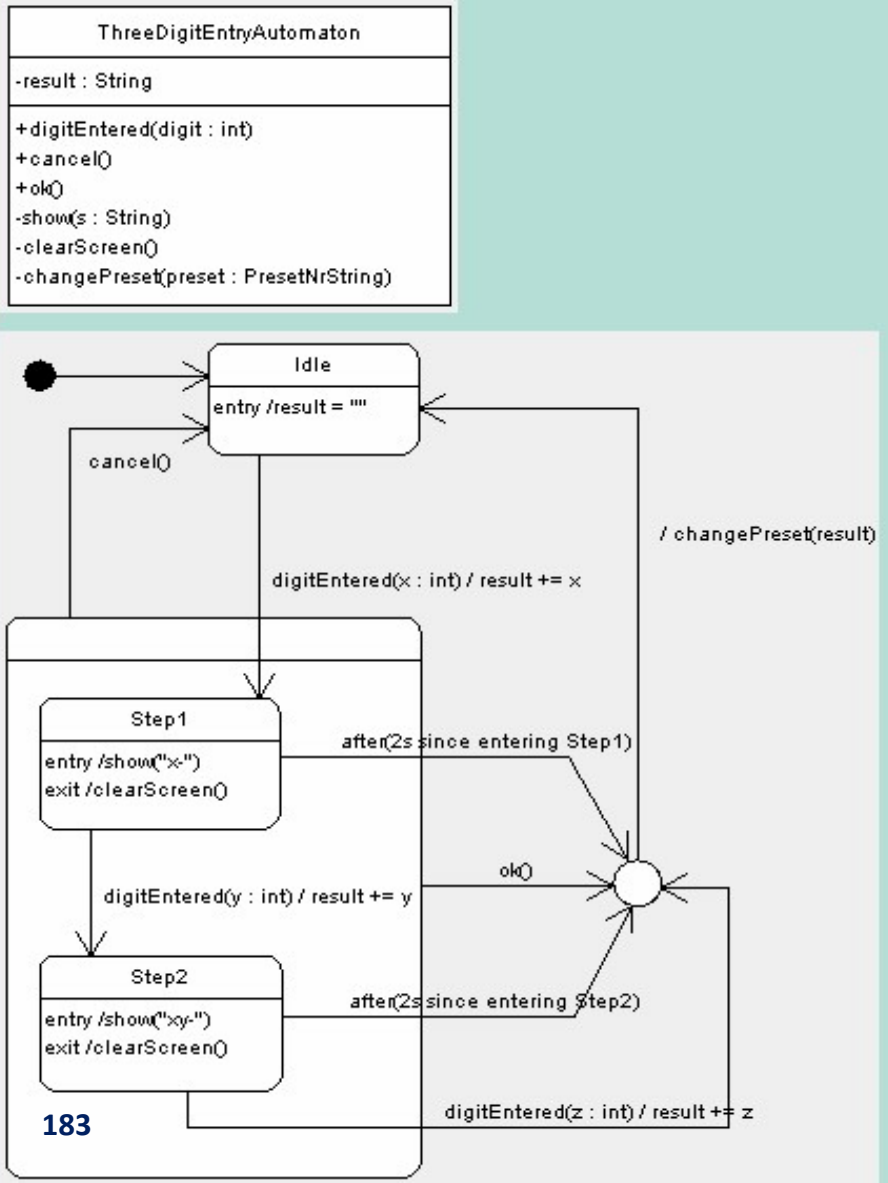
The UML diagrams

Class diagrams

Example

Legend:
digitentered(int: nr): This event is triggered when a RC digit key is pressed; the parameter stands for the number.
cancel: This event occurs when e.g. the P+/P- keys are pressed.
ok: This event occurs when the RC_OK key is pressed.
show(string): This action displays the given string in the Digit Entry OSD (as defined in item 9712).
clearscreen: This action removes the Digit Entry OSD (as defined in item 9712) from the screen.
changePreset(result): This action comprises checking if the "result" is representing a valid preset number, and acting accordingly.

9767 The user shall be able to select tuner presets using "3 digit entry", which works as follows:



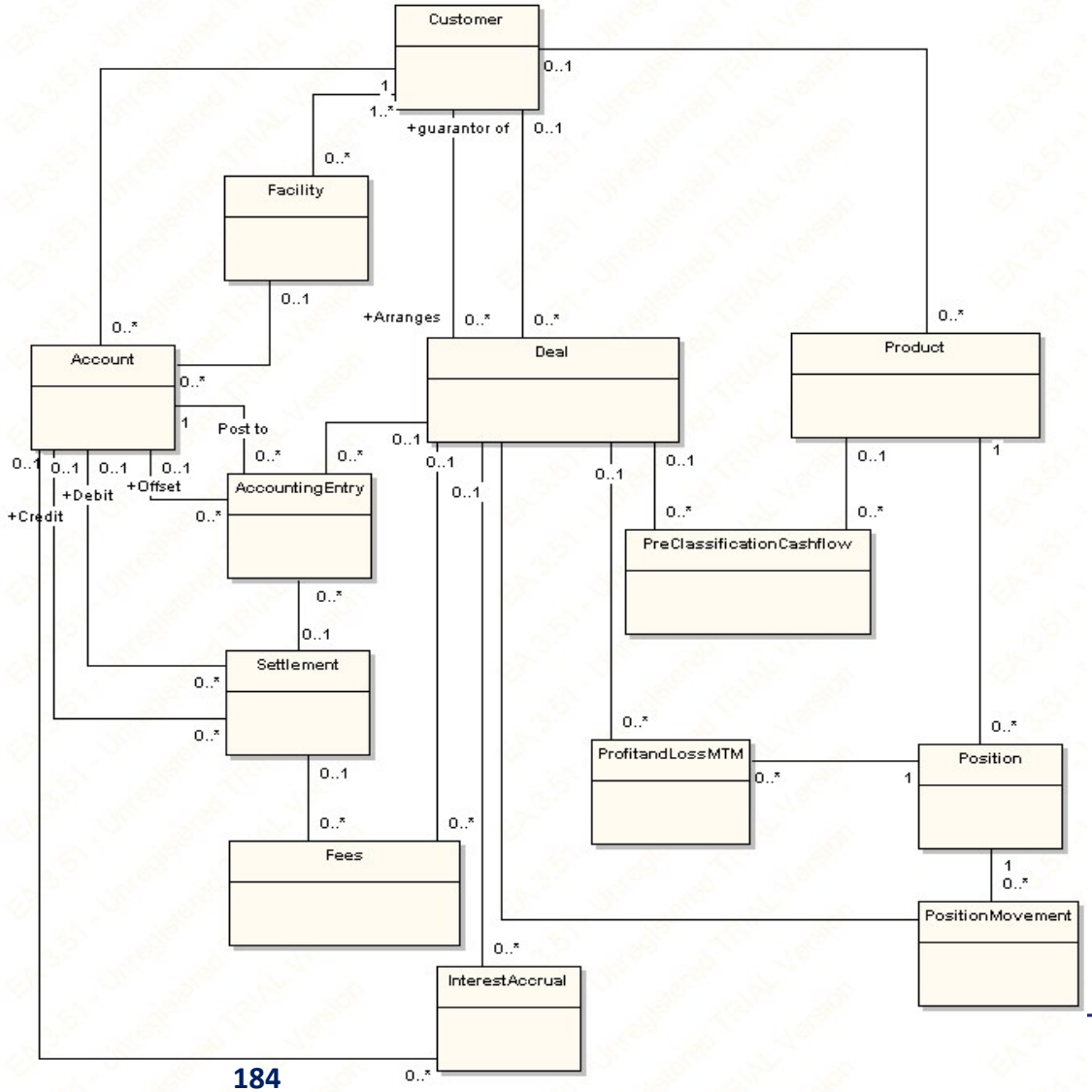
SYNTAX OF UML

The UML diagrams

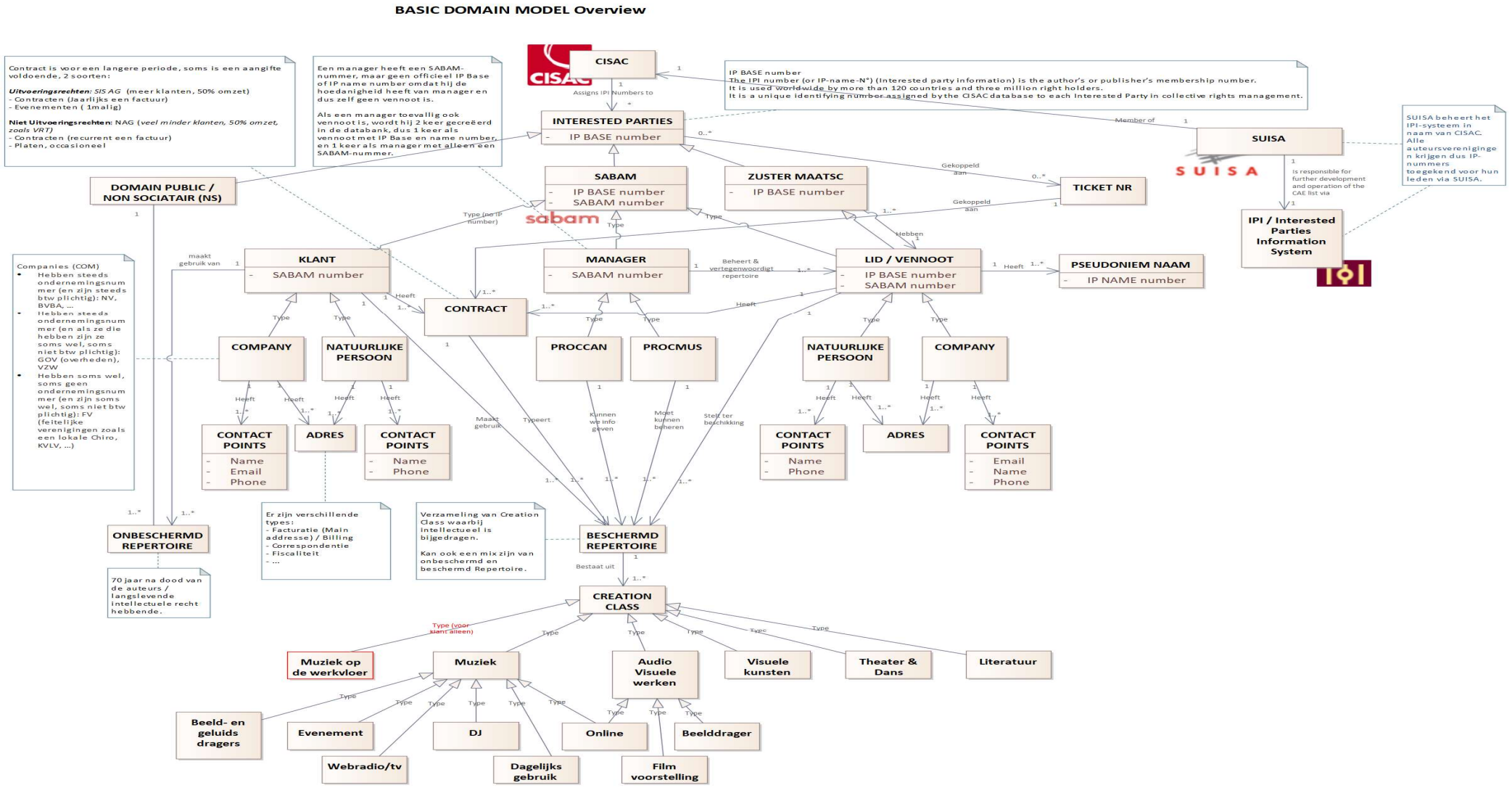
Class diagrams

Example

Customers & deals



UML - Examples



Business Analyse Technieken: modelleren en realiseren van requirements

32 (0) 476 99 59 93

www.Gijsels.com

Christian@Gijsels.com

https://www.facebook.com/TheInstituteBelgium

http://www.linkedin.com/in/christiangijsels

https://twitter.com/gijselsdotcom/

NCOI LEARNING

GIJSELSDOTCOM CONSULTING | the Partner for your Digital Transformation www.gijsels.com