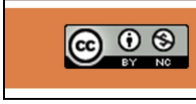
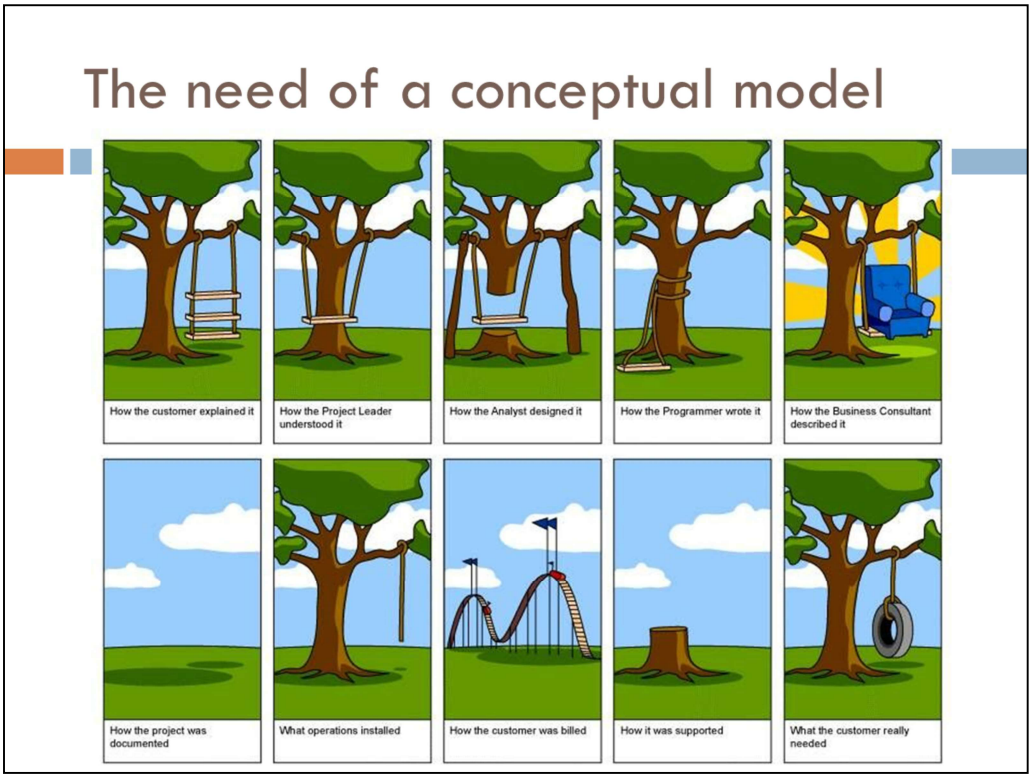
SIMULATION MODELS CONCEPTUALIZATION



Pau Fonseca i Casas; pau@fib.upc.edu

We are going to discuss now how we can define a Simulation model conceptually. On this group of slides, we will see the characteristics that a conceptualization of a simulation model must own.



This is a well-know funny picture that shows some of the problems, related mainly with the communication, that can happen in the process to develop a new software.

It is obvious that this can be funny but,… if this is not your project. Also, in a simulation project, where the final result will be a product that we will use to make decisions, this becomes crucial.

Hypotheses

- What is inside the model?
- Hypotheses
  - Systemic / Structural
  - Simplification

But how we are going to define and communicate the different key elements of our model?.

The main element to understand, is that our simulation model, like any other model, is working with assumptions. This assumptions are based on the knowledge that the experts have, and on the data and observations we can use. Mainly the type of assumptions we can use are simplification assumptions, that are those assumptions that are not true, but that are needed in order to constrain the time, or the resources, needed to define the model, or because the knowledge we have is not enough to define a more detailed model. The Systemic assumptions, can be divided on structural and data assumptions, depending if these assumptions are related with the data we will use in or model, or with the structure that the different elements that we will use in our model.

Advantages of use a conceptual model

- Textual specification is less precise.
- Conceptual model have in a detailed manner, the dynamical relations between the different elements of the interest process.
  - Constitutes an specification by itself.
- Simplifies the dialog between the different parts that are involved in the project.
- Constitutes a representation of the simulation model independent of the selected tool used to build the model.

To address the need to define clear and concise the model assumptions, we will use a conceptual model.

A textual representation is not precise, since natural language owns an inherent ambiguity, hence we will avoid the use of this method, although there are some methods that rely on it, like ODD protocol

The conceptualization of the model must own in a detailed manner, the dynamical relations between the different elements that the modeler consider of the interest for the process definition. Because the complete and unambiguous nature of the conceptualization languages we will learn, this constitutes a specification by itself.

Is important the conceptual model simplifies the dialog between the different parts that are involved in the project. This is a key aspect, since we want to introduce all the parties involved in the project in the model development. As we will see in Validation, Verification and Accreditation part, this is a key element to assure that the model achieves its goal, it is used for decision making.

An important point to note is that the conceptual model constitutes a representation of the simulation model independent of the selected tool used to build the model. That means that is not constrained to a specific codification tool.

Also, it is worth to mention that conceptual models can be considered

analytical models if they can be solved analytically or numerically, like as an example Queue Models.

Conceptual model formalization

- Formalism must be **independent** from the simulation tools.
- The formalized model must **allow** some **analysis.**
  - To **determine relations** between components.

To summarize two of the important point of the previous slide, we note that, the conceptual language selected must be independent of the simulation tools used. Also, it is strongly desirable that the formalized models allows some analysis. We will see that the approaches we present allows this analysis, like the determination of deadlocks or the understanding of the steeps needed to achieve an specific state. Mainly thought the analysis of the relations that exists between the different components.

Conceptual model formalization

- Formalism must allow an easy transformation to the representations supported by the existing simulation frameworks.
  - Simplify the codification process.
  - To evaluate alternatives.

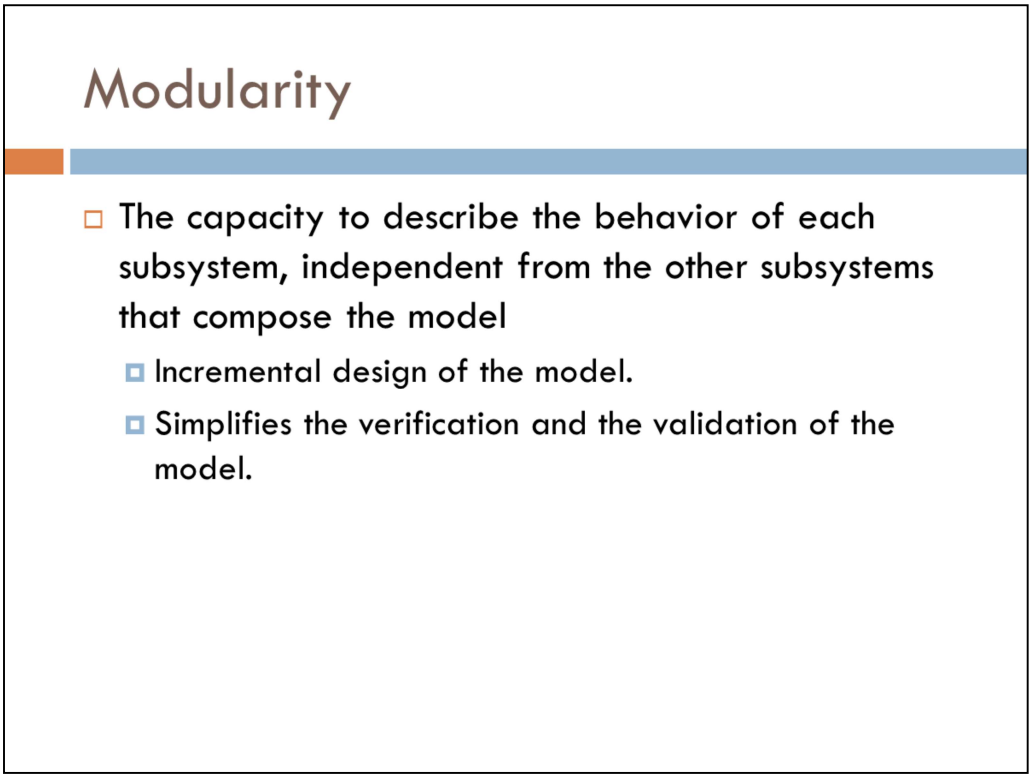
The formalism we will use to define the conceptual model must allow also an easy transformation to the representations supported by the simulation frameworks that we will use in our project, that means, the simulation tool we will use to do the specific codification. This will simplify the codification process simplifying the Verification process that one must do.

Also, since simplifies the re-codification, we can evaluate different alternatives, that means, different codifications of our simulation model.

Conceptual model formalization

- Some aspects of the model can be no specified, without causing problems in the transformation to other representations. MODULARITY
- The model must be defined in terms that **no constrain** its codification in a **particular mechanism** of simulation **clock update.**

Two important concepts that a conceptual model formalization must be assured, are the MODULARITY and the freedom to select the method to update the clock that the simulation model needs, like the one followed by Event Scheduling that differs from the Activity Scanning approach.



Modularity is defined as the capacity to describe the behavior of each subsystem, independent from the other subsystems that compose the model

This allows an Incremental design of the model, since we can start defining a very simple modules and increase its complexity depending on the need of our analysis.

Also this simplifies the Verification and the Validations, since we can be focused in each one of the different modules of our model and perform this Validation and Verification processes independent for each one of them. Since the models are representing lust a part of the overall module this Validation and Verification process is simplified.

Assure the Modularity

1. A module cannot access directly to the state of other modules or components.
2. A module must own a set of ports (input/output) to allow the interaction with the other parts of the model.

To assure the modularity one must follow this rules. First, a module cannot access directly to the state of other modules or components of the model. That means that if we are defining a queue system, that is composed by a queue and a server, the server cannot access directly to the queue elements or stated variables. Form the point of view of the server the queue is a black box, and vice versa.

Secondly, since we need to stablish a communication between the different modules to exchange the information, a module must own a set of ports (input/output) to allow the interaction with the other parts, modules, of the model.

Conceptual models

- Flow models.
- Queue networks.
- BPM.
- Service Blueprint.
- Petri nets.
- Colored Petri nets.
- SDL language
- DEVS
- Causal and Forrester diagrams.

There are several ways to define a conceptual model, here we have some of the more usual Methods to define a conceptual model in simulation.

We will review some of them and we will discuss and review in detail the more powerful to be used in the simulation scope.

Working with different formal languages

- Three of the main mechanisms for doing this:
  - Meta-formalism.
  - Common formalism.
  - Co-simulation.
- Vangheluwe, H. L. (2000). DEVS as a common denominator for multi-formalism hybrid systems modelling. *IEEE International Symposium on Computer-Aided Control System Design* (pp. 129--134). IEEE Computer Society Press.

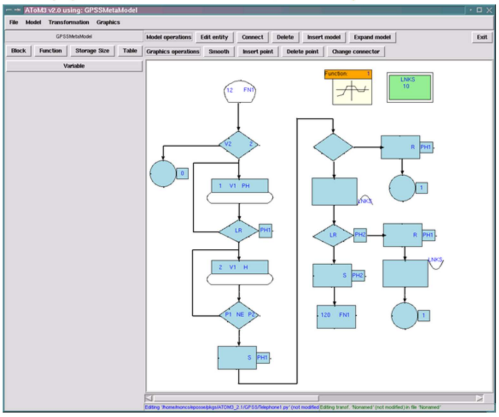
At this point is also interesting to mention that the selection of the formal language that we will use to do this conceptualization is relevant, but not really important, since if we are using one of them, that assures modularity and also the independence of the clock used to update the time, it will be relatively easy to combine different simulation models, conceptualized with different formal languages in the same environment.

The three approaches we can use are Meta-formalism, Common formalism, and the last one that is Co-simulation, that allows also to combine simulation models that lacks the formal representation, that means that are only represented by tis codification.

On this paper you can see how, one can transform from one conceptualization to other. Notice that although this is possible, not all the transformation paths between the different formalism has been described, hence depending on our need, maybe we must define a new mechanism that fits our needs.

Meta-formalism

- A formalism that incorporates the different formalisms of the various sub models that makes up the system.
- ATOM3: http://atom3.cs.mcgill.ca/

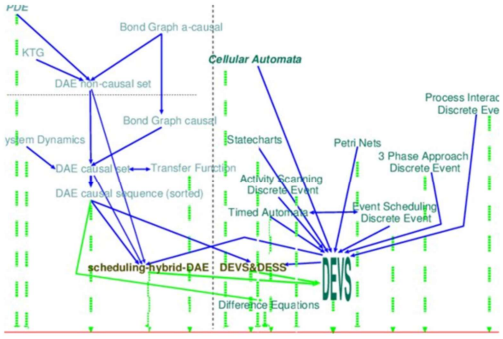


Meta-formalism approach is a mechanism that incorporates the different formalisms of the various sub models that makes up the system. As an example, one model can be defined in Petri Nets, while other in SDL, and Meta-formalism approach allows to combine all this formalism in a single simulation.

On system that follows this approach is ATOM3.

Common formalism

- A mechanism that converts all formalisms to a common formalism.
- Transforming algorithms from:
  - SDL→DEVS→Petri Nets…



https://www.researchgate.net/publication/220713394_Computer_Aided_Multi-paradigm_Modelling_to_Process_Petri-Nets_and_Statecharts

Common formalism approach is based on the idea that we can transform all the representation of a simulation model to other representation, as an example, is you have a model defined in Petri Nets you can transform it to state charts or DEVS to use this model I a biggest model.

On the paper you can read regarding this approach.

Co-simulation

- Independent simulators that work together
- HLA: The **High-Level Architecture** (**HLA**) is a general-purpose architecture for distributed computer simulation systems. Using HLA, computer simulations can interact to other computer simulations regardless of the computing platforms. The interaction between simulations is managed by a Run-Time Infrastructure (RTI).

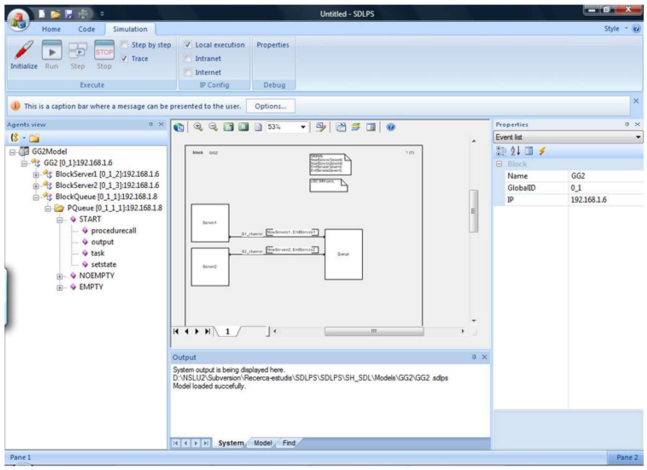
Finally, co-simulation approach is a technique that can be used to include in the model legacy simulation models, that lacks a proper conceptualizations,, that means, the models where you only have the codification.

One widely used approach to this is High Level Architecture (HLA). It is a general-purpose architecture for distributed computer simulation systems.

With HLA, computer simulations can interact ,that is, to communicate data, and to synchronize actions, to other computer simulations regardless of the computing platforms. The interaction between simulations is managed by a Run-Time Infrastructure (RTI). This method is now a IEEE standard and was developed in the 90's under the leadership of the US Department of Defense.

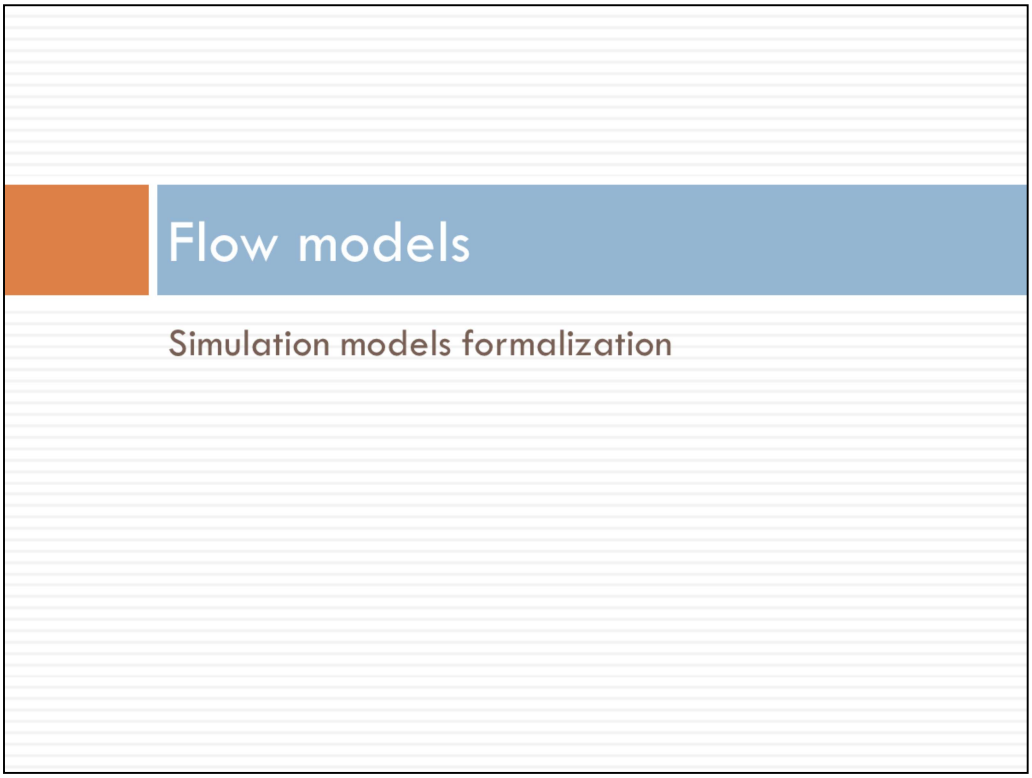
Co-simulation with SDL

- We use SDLPS (on the practical sessions)



https://sdlps.com/

In UPC we own a simulation infrastructure, named SDLPS that allows to execute a simulation model represented in SDL and to interact with other simulation engines following a Co-simulation approach.



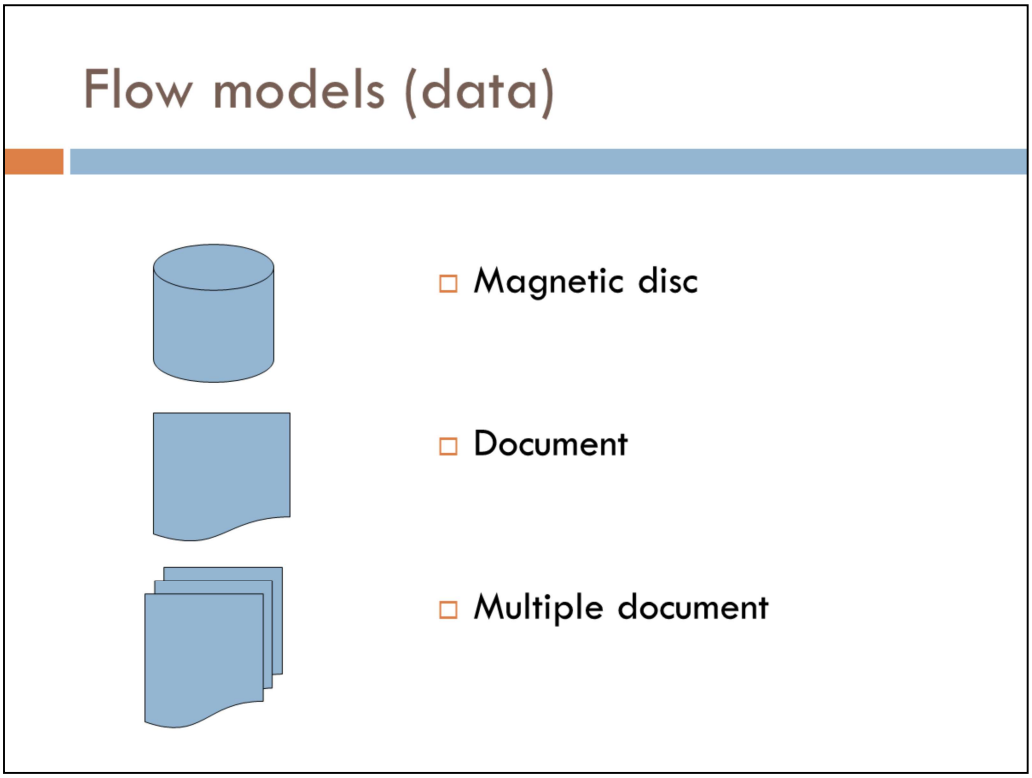
Now we are going to review some techniques to define formally a Simulation model. We will start with Flow models.

Flow diagrams

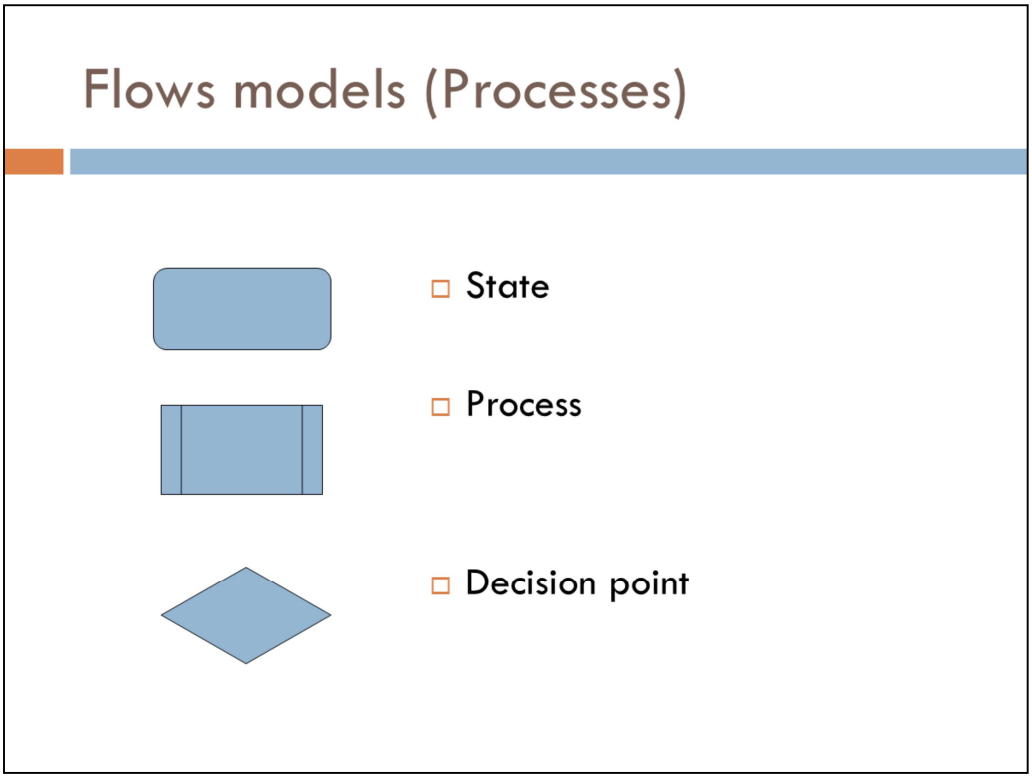
- Flow chart or flow diagram... is a diagram that visually displays interrelated information such as events, steps in a process, functions, etc., in an organized fashion, such as sequentially or chronologically.
- Flow diagram [is] a graphic representation of the physical route or flow of people, materials, paperwork's, vehicles, or communication associated with a process, procedure plan, or investigation.
- See *Information Graphics: A Comprehensinvestigationive Illustrated Reference* by Harris (1999)

https://en.wikipedia.org/wiki/Flow_diagram

Here you can view the definition for a flow diagrams. Please read it. Although there are several different types of flow diagrams, we will concentrate on the more generic approximation to define flow models for simulation.



Flow models are based on diagrams that joins several icons that represents some main elements of our System. In this char are presented some of the more common elements one can use, like a Magnetic disc, a Document or Multiple Document.



On this slide we show the icons that usually represents the processes, like the State, a Process and a decision point. Notice that there is not an standard that defines exactly how this elements must looks like, and mainly this is based on a convention.

Pediatrics example

- Models a pediatrics example.
- If a new emergency arrives a special process takes cure of it.
- If X ray is needed, or blood analysis, is done in a second visit
- Finally the patient release the system.

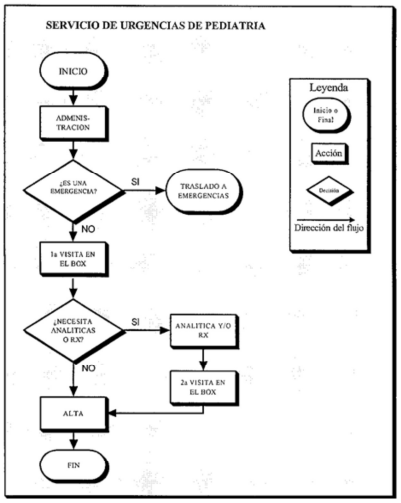
We are going to model a very simple example, a huge simplification of a pediatrics urgency center.

The process follows this different steeps, first, if a new emergency arrives a special process takes cure of it. We didn't model this.

Then if a normal visit enters, we see if is needed a X ray or blood analysis. If this is the case, a second visit is planned, if no the patient leaves the system happy.

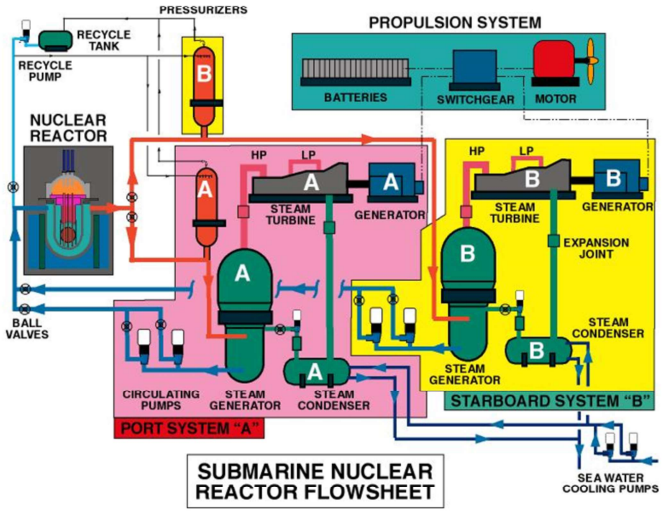
Tif a second visit is planned the patient returns and finally leaves the system healthy.

Flows models



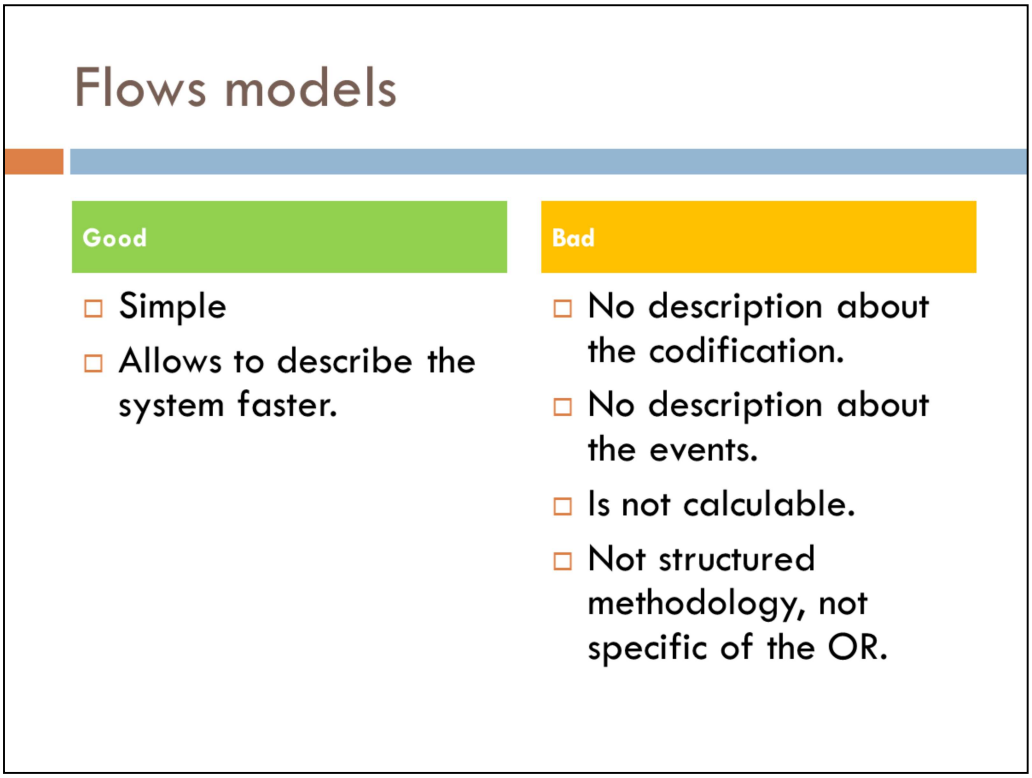
This can be the representation in a flow model of this System. Can you understand all the processes?

Submarine nuclear reactor



"SUB REACTOR SYSTEM FLOW". Licensed under Public Domain via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:SUB_REACTOR_SYSTEM_FLOW.jpg#mediaviewer/File:SUB_REACTOR_SYSTEM_FLOW.jpg

And what about a flow diagram that shows the behavior of a submarine nuclear reactor?



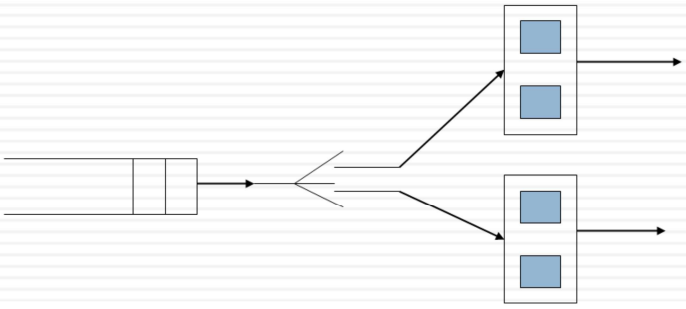
Here you have a list of the best and worse characteristics of flow models.

The main problem of this approach is that it don't detail what happens with the evets that rules the model behavior, this make really difficult to obtain a codification, or a set of codifications) that obtains exactly what is intended to be represented on the diagrams.

This is not a Operations Research (OR) methodology, and we will not use it.

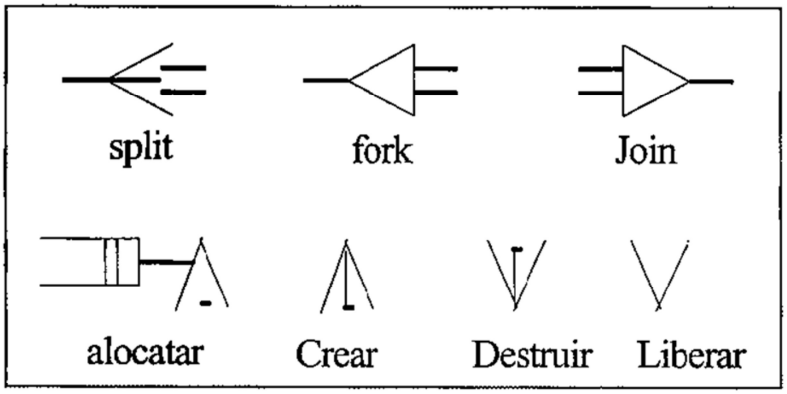
Queue networks

Simulation models formalization

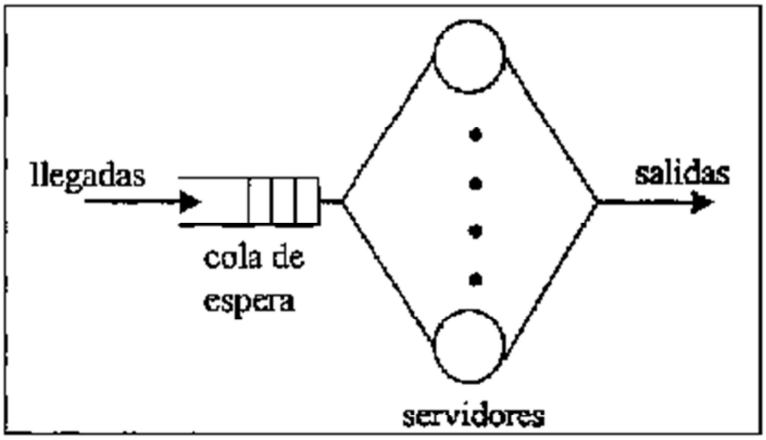


Basic structure of queue models

Queue networks

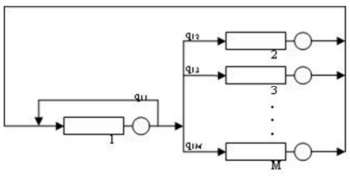


Queue networks (M|M|S)



Queue networks (best)

- Simple
- Allows to understand the system faster.
- Specific to describe queue models.



Queue networks (worse)

- No description about the implementation.
- Do not describe too much about the events management.
- Is not always calculable.
  - Some models can be calculated following the queue theory.

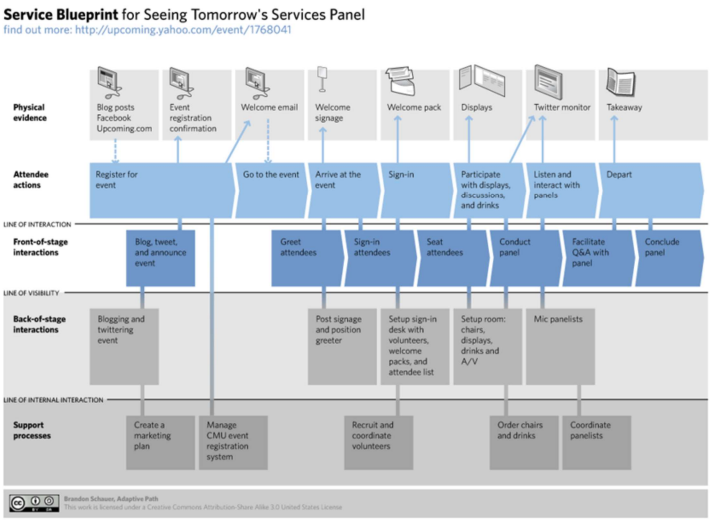
Service blueprint

From the customer perspective

Service Blueprint

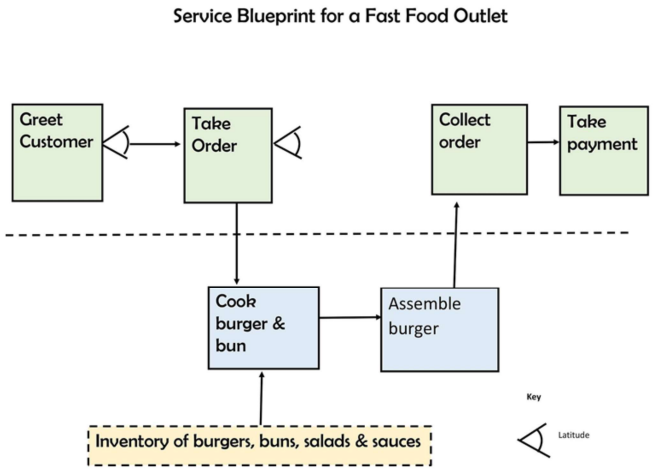
- The **service blueprint** is a technique originally used for service design and innovation but has also found applications in diagnosing problems with operational efficiency.
- The technique was first described by G. Lynn Shostack, a bank executive, in the *Harvard Business Review* in 1984.[1]
- The service blueprint is an applied process chart which shows the service delivery process from the customer's perspective. The service blueprint has become one of the most widely used tools to manage service operations, service design and service positioning.

Service Blueprint

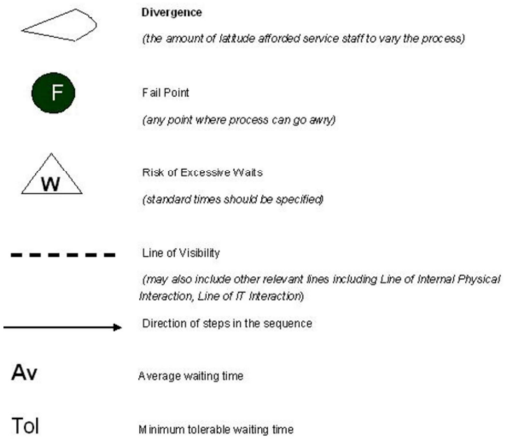


By brandon schauer - https://www.flickr.com/photos/brandonschauer/3363169836, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=41292602

Service Blueprint



Blueprint common simbols



Divergence

(the amount of latitude afforded service staff to vary the process)

Fail Point

(any point where process can go awry)

Risk of Excessive Waits

(standard times should be specified)

Line of Visibility

(may also include other relevant lines including Line of Internal Physical Interaction, Line of IT Interaction)

Direction of steps in the sequence

Average waiting time

Minimum tolerable waiting time

BPM

Business Process Modeling

BPML

Business Process Modeling Language

BPML

- BPML was designed as a formally **complete language**, able to model any process, and, via a BPMS (business process management system), deployed as an executable software process without generation of any software code.
- This is not possible with BPEL, since BPEL is not a complete process language. In practice BPEL is often used in conjunction with Java to fill in the "missing" semantics. In addition, BPEL is often tied to proprietary implementations of workflow or integration broker engines. Whereas, BPML was designed, and implemented, as a pure concurrent and distributed processing engine. It was designed to be semantically complete according to the Pi-calculus formal representation of computational processes.
- BPEL and BPML are examples of a trend towards process-oriented programming. BPEL and BPML herald the concept of a BPMS as an IT capability for management of business processes, playing a role similar to a RDBMS for business data.

BPEL

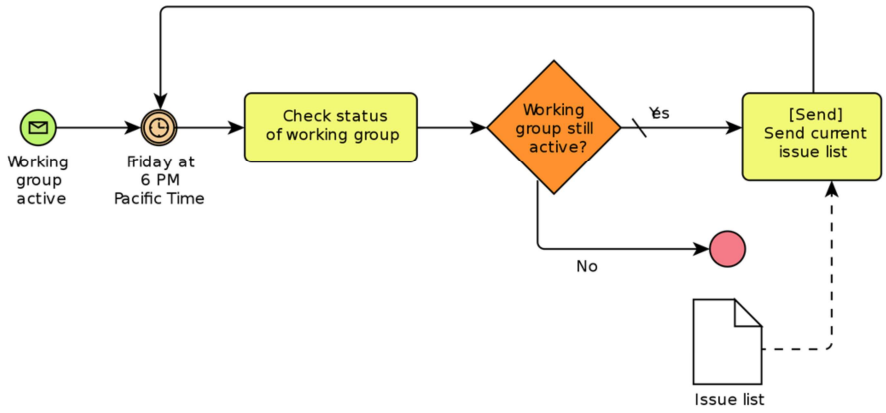
Business Process Execution Language

- The **Web Services Business Process Execution Language** (**WS-BPEL**), commonly known as **BPEL** (**Business Process Execution Language**), is an OASIS[1] standard executable language for specifying actions within business processes with web services. Processes in BPEL export and import information by using web service interfaces exclusively.

BPMN

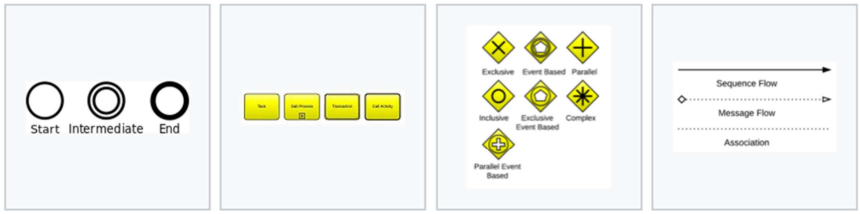
Business Process Model and Notation

BPMN



By BPMN-AProcesswithNormalFlow.jpg: Tttt1234derivative work: Hazmat2 (talk) - This file was derived from: BPMN-AProcesswithNormalFlow.jpg:, Public Domain, https://commons.wikimedia.org/w/index.php?curid=18126163

BPMN



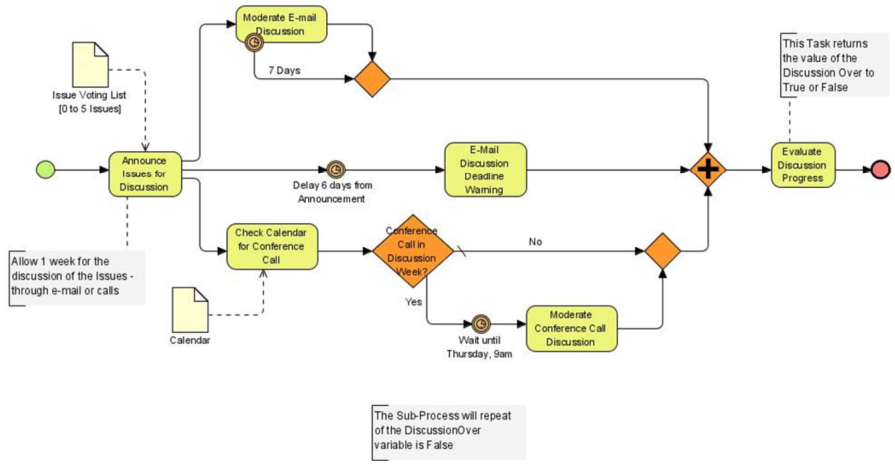
Event Activity Gateway Connections

By serol1971 - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=25506270

Swim lanes and artifacts

Swimlanes | Data objects | Groups | Annotation

Discussion cycle BPMN example



By Ttt1234 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=4845066