Redes de Petri

Modelling systems and Petri nets

- systems are organizational entities
  - regulated flows of objects and information are significant
- Petri nets have proven to be very reliable in practice for the design of such systems
  - simple and immediately understandable principles
  - provide an illustrated description without having to go into the mathematics behind it
  - first edition (in German) appeared in 1985.

Software engineering

- In the 1960s, software developers worked in blackboard
- wide variety of sketches: informal box diagrams, flow charts, transition nets, state diagrams and many others
- illustration of problem analyses or specifications
  - People understood each other more rapidly and better in discussions
  - Meetings with users were more pleasant and more effective
  - Specifications and documentations were less tiring and more easily understandable.

- Things have changed since then. The disappearance of blackboards has led to the disappearance of diagrams in most presentations, specifications and software descriptions.
- What is behind this change (for the worse presumably)?
    - The batch computing situation developed into on-line programming.
    - an elimination of waiting times and blackboards. They were replaced by terminals "unable to draw"
- In the early 1970s "structured programming" and "stepwise refinement" methods gave rise to a new software technology.
- in the early 1980s, university professors maintained that Petri nets were the same as flowcharts

About Petri nets

- Petri netswere already around in the early 1970s.
- Software systems have now been developed that permit the user to represent Petri nets graphically, store them, edit them, check them and revise them.
- Channel-agency nets are "structured".
- Their graphic syntax is precisely defined
- Semantics exist for them

PN-based design methods

- Petri nets is a straightforward method, that it is possible to fall back on formalisms
- An important function of any software engineering method is the support of coordination and information exchange between users, programmers and those persons or organizations who request program development.
- a very practical design method that can help solve one of the central problems of application-oriented software development - the improvement of communication between the parties involved.

Petri Nets: Why, What, When, How - Part I

Why Petri nets have been and are being used in many areas of data processing for the modelling of hardware, communication protocols, parallel programs and distributed data bases, particularly in the requirements engineering context, i.e. in the initial phase of system design until implementation and execution.

What organizational, i.e. logistic, technical and computer-integrated systems of all kinds in which regulated flows of objects and information are of significance.

Petri Nets: Why, What, When, How - Part II

When a system is being planned or an existing system analyzed, it frequently is the case that the system is incompletely and unclearly described and, in addition, a software customer is involved in planning an analysis but is unable to handle formal representations.

to break the system down into rough components, the functions and interrelationships of which can be described in detail. Dynamic behavior is defined more and more in the design phase as well. The overall design process must be systematically documented.

How is it possible to clarify system design questions in a non-formal and yet precise manner.

an informally or incompletely described real system be broken down in a meaningful manner and its components individually analyzed.

Petri Nets: Why, What, When, How - Part III

we move from informal system representations to descriptions that can be used as a basis for formalization

we handle systems that are hierarchically structured or are made up of several components that are relatively independent of one another

we separate different views and relate them to one another at different levels of abstraction

we model the (dynamic) behavior of a system alongside its (static) structure at the highest possible level

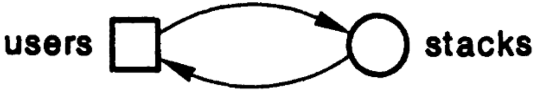
we represent individual and interrelated processes in (distributed) systems

Alternatives

- procedures that systematize the route from the formulation of the problem to a viable program. The initial phases of such procedures are devoted to the assessment and formulation of system requirements. This is referred to as "requirements engineering".
  - Petri nets give the same treatment to active and passive components and, above all, they provide the possibility of progressing, at a high level and with any degree of precision, from the description of static components to the representation of dynamic behavior.

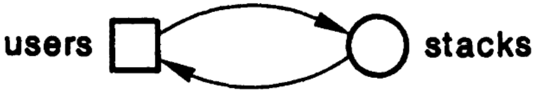
Example: library

- The users of the library have access to a stock of books kept in the stacks.
- It would seem obvious enough to describe one of the components (the stock of books) as being passive, and the other (the users) as being active.
- The arrows between the components indicate the flow of objects and information.



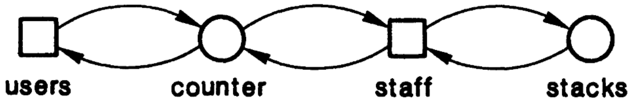
Example: library

- In large libraries a user may not have free access to the stacks.
- Instead there may be one (or several) counters at which users are served by library staff.
- Intuitively we understand which components are passive (the counters) and which are active (library staff).



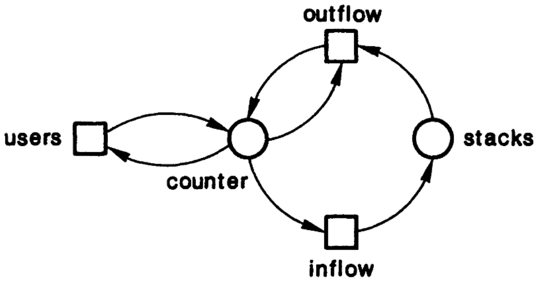
Example: library

- In large libraries a user may not have free access to the stacks.
- Instead there may be one (or several) counters at which users are served by library staff.
- Intuitively we understand which components are passive (the counters) and which are active (library staff).



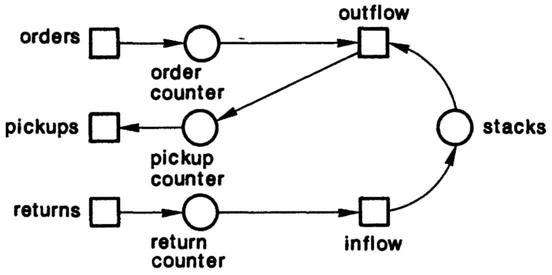
Example: library

- A large volume of book borrowing requires organization. We first of all need to distinguish the lending (outflow) and return (inflow) of books.
- We assume that before books can be borrowed they first of all have to be ordered.



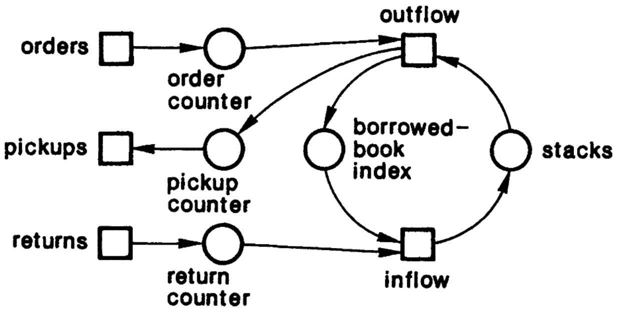
Example: library

- separate counters for ordering, picking up and returning books
- The library keeps a record of the books borrowed, in the simplest case by using index cards that are stored with the books in the stacks.
- Whenever a book is borrowed, its card is placed in a borrowed-book file.



Example: library

- When the book is returned, the relevant card is taken out of the file and returned to the stacks along with the book it belongs to.
- Thus, the net adds a borrowed-book file to the system.

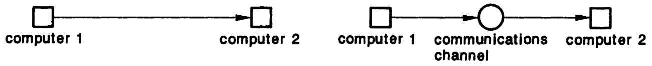


Distinguishing components

- The first thing that needs to be done is to distinguish components of the system and to designate every component as being either passive or active.
- The passive components (i.e. the various counters, the stacks and the borrowed-book index) are able to store things or make them visible. They can be in different states. They are referred to as channels.
- The active components (i.e. ordering, picking up and returning books, as well as the act of retrieving books from the stacks and the act of putting them) are able to produce, transport or change things. They are referred to as agencies.
- An arrow never represents a system component, but rather always an abstract relationship between components, e.g. logical connections, access rights, physical proximity or direct links.

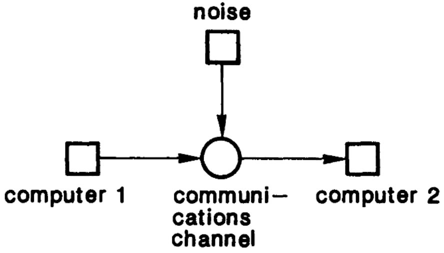
Linking components

- there are no arrows connecting any two passive or any two active components (i.e. two channels or two agencies) with one another. Instead, each arrow leads from a channel to an agency or, conversely, from an agency to a channel.

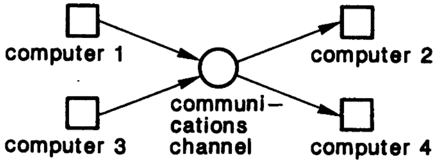


- a wrong and a right way (i.e. in terms of Petri nets) of modelling the communications channel in a computer link. In the drawing on the left the communications channel is either not modelled as a separate component, or it has been assumed that channels of this kind always connect two computers.

- modeling actions of this kind.

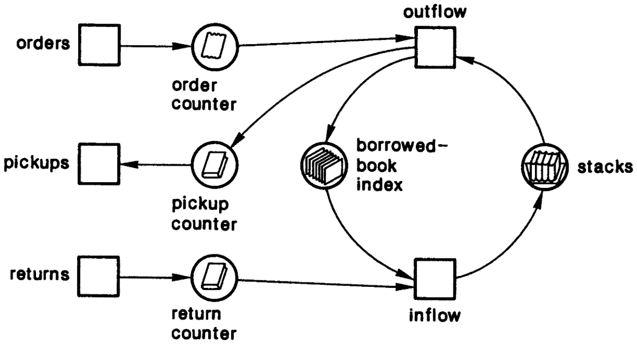


- linking several computers via a single communications channel



- the channels contain concrete objects, i.e. an order form (filled out) is at the order counter, a book is at the pickup counter waiting for the person who ordered it, and at the return counter there is a book that has been returned but not yet put back on the shelf.
- The books in the stacks and the cards in the borrowed-book index are also indicated.
- The agencies can now redistribute the objects, acting in accordance with rules formulated by the system designer. Order forms are accepted by the lending agency, books are retrieved from the stacks and brought to the pickup counter.
- Borrowers pick up the books they have ordered from this counter.

Objects (Tokens) into PNs



Synthesis of principles to PN-based modeling

- breaking the system down into passive and active components,
- the transition from the static nature of separate components to the dynamic behavior of a system
- the interrelating of individual net representations

Comparative among PN tools - Part I

	Pipe 2	CPN Tools	HP Sim
Petri Net Types	Basic, stochastic,colored, timed andhierarchical	Basic, colored,timed, hierarchical	Basic, stochastic,timed
Components	Graphical editor, token game, graphical simulation, statespaces, Invariantplace, structuralanalysis, simple performance analysis, interchange fileformat, extensive modulus analyses, archive format analyses	Allows user code, graphical editor, Fast Simulation, token game animation, statespaces, simple performance analysis, interchange file format	Graphical editor, tokengame animation, fastsimulation, simple performance analyses
Environments	Java	Linux(desempenho inferior), Windows	Windows
Simulation Results	Simple Results	Simple and Complex results	Simple results
Editor	Zoom, exportation,editing, Auto-adhesive notes, undo, redo	Zoom, editing,animation, undo,redo, cloning	Zoom, print function, text annotations,editing