Business Processes Modelling

MPB (6 cfu, 295AA)

Roberto Bruni
http://www.di.unipi.it/~bruni

01 - Introduction


Witaj
Gabitê
Welkom
Velkommen
benvenuto
Welcome
Bem-vindo
Välkommen
Bienvenue
Καλώς ήλθατε
Maeva
Marhaban
Selamat
歡迎
में आपका सवागत है
Vítejte
Welkom
Hosgeldiniz
Yôkoso
Bienvenido
добро пожаловать
Willkommen

Every

Tuesday:
11:00-13:00
M1

Thursday:
16:00-18:00
L1

Course Material shared on Microsoft Teams and DidaWiki

No streaming / recording planned

Who am I?

http://www.di.unipi.it/~bruni

bruni@di.unipi.it

Office hours: by appointment
preferably
Wednesday 16:00-18:00

Who are you?

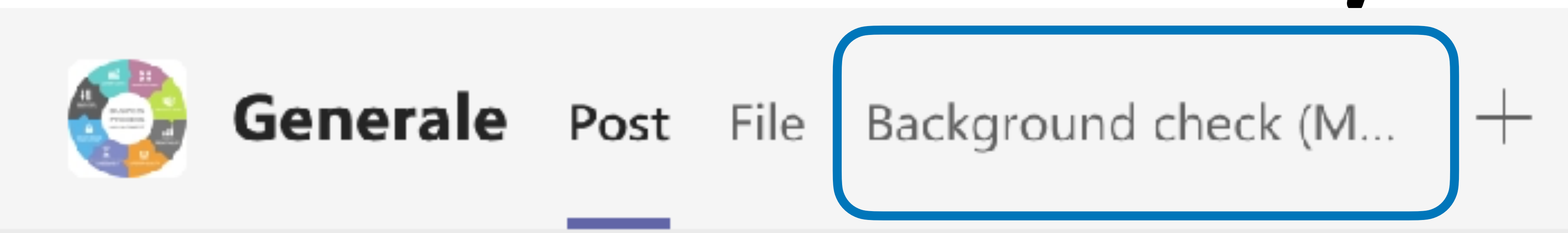
Hi, I’m … from …

I have studied … at the University of …

I am now a student of the MSc degree in …

Hi, I’m Paolo Rossi from Italy
I have studied Computer Science at the University of Pisa (Italy)
I am now a student of the MSc degree in Data Science and Business Informatics

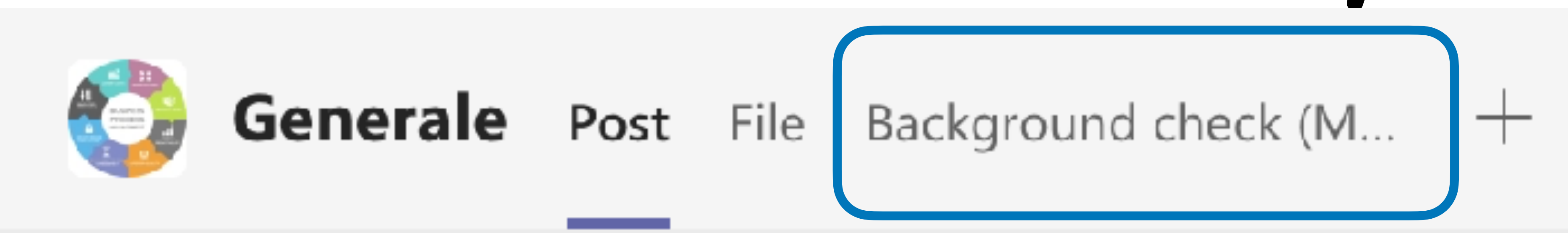
Who are you?



Background check (form)

- Enrollment number:
- Bachelor degree:
- MSc course of enrollment:
- Subjects of interest:
- … :

Who are you?



Background check (form)

Enrollment number: 123456

Bachelor degree: Comp. Sci., Pisa, IT

MSc course of enrollment: Data Science and BI

Subjects of interest: Data analysis, AI

… :

The Course

Some quotes

All models are wrong, but some are useful
- George Box

Subjects are divided in two categories:
1) *too difficult matters, that CANNOT be studied*
2) *easy matters, that DO NOT NEED to be studied*

- back of a t-shirt

What is a BP?

Any set of steps aimed to reach some outcome

from opening an account

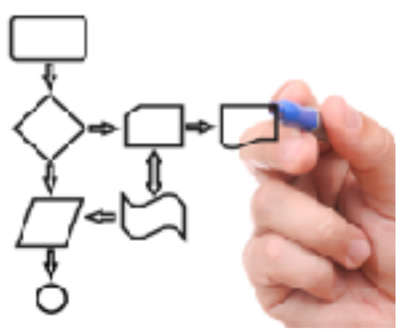
to processing a custom order

What is BPM about?

Course objectives

Key issues in Business Process Management (patterns, architectures, methodologies,…)

Analysis techniques and correctness by construction (soundness, boundedness, liveness, free-choice,…)



Graphical languages & visual notation (BPMN, EPC, ...)

Tool-supported verification (WoPeD, ProM, Woflan, ...)

Structural properties, behavioural properties and problematic issues (dead tasks, deadlocks, ...)

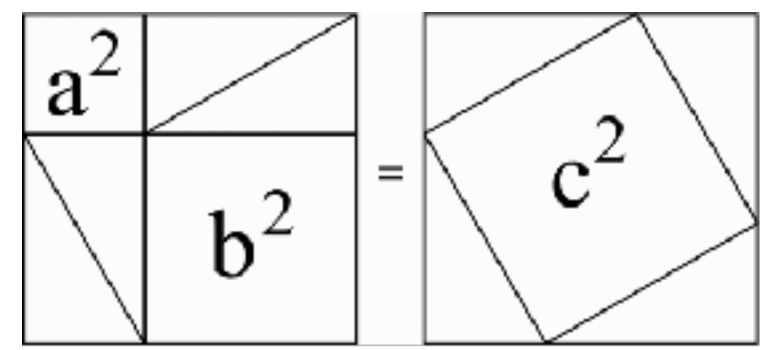
Performance analysis (bottlenecks, simulation, capacity planning,…)

Formal models (automata, Petri nets, workflow nets, ...)

Process mining (discovery, conformance checking, enhancement,…)

Course activities

attend classrooms:
ask questions!
(sleep quietly)



learn theorems:
(drink many coffees)

do some thinking:
solve ALL exercises
(at least try to)

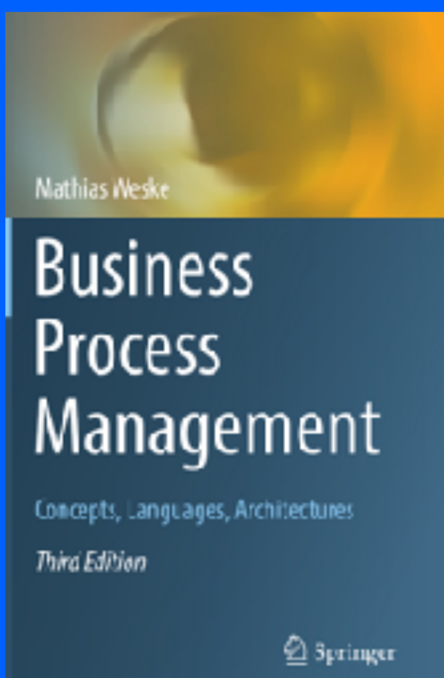
deliver a project:
practice with concepts,
experiment with tools

give the exam:
time for a party!

Exam

- Project + Oral exam (see FAQ for more infos)
- Group project
  - at most 2 students for group
  - can be requested any time
  - three-weeks deadline
  - modelling + analysis + short report
  - the validity of the project is for the current academic year
- Oral exam
  - registration mandatory
  - actual date fixed at the time of project delivery
  - group discussion of the project
  - followed by individual questions on course topics

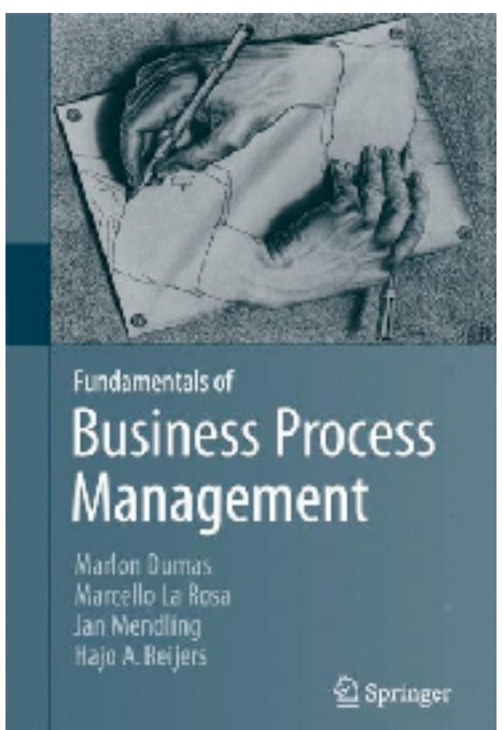
Main Textbook



Mathias Weske
Business Process Management: Concepts, Languages, Architectures
(3rd ed.) Springer 2019
http://bpm-book.com

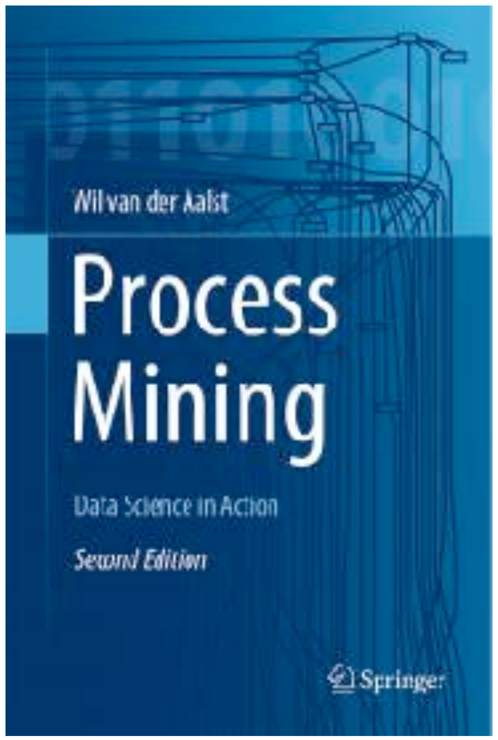
Suggested Reading

Joerg Desel and Javier Esparza
Free Choice Petri Nets
Cambridge Tracts in Theoretical Computer Science 40, 1995
https://www7.in.tum.de/~esparza/bookfc.html

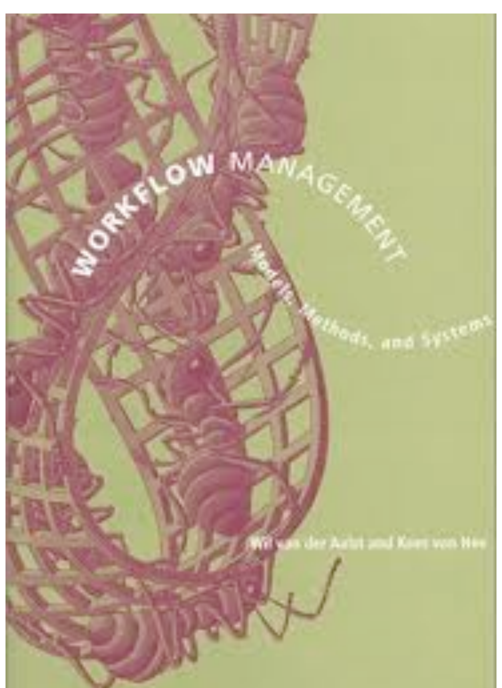


Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo Reijers
Fundamentals of Business Process Management
(2nd ed.) Springer 2018
http://fundamentals-of-bpm.org

Other Textbooks



Wil van der Aalst
Process Mining
(2nd ed.) Springer 2016
http://springer.com/978-3-662-49850-7



Wil van der Aalst, Kees van Hee
Workflow Management: Models, Methods, and Systems
MIT Press (paperback) 2004
https://mitpress.mit.edu/books/workflow-management

Main resources

Petri nets world

http://www.informatik.uni-hamburg.de/TGI/PetriNets

BPMN

http://www.bpmn.org

Workflow Patterns

http://www.workflowpatterns.com

Main resources (tools)

WoPeD

http://www.woped.org

ProM (and Woflan)

http://www.promtools.org/

http://www.win.tue.nl/woflan

Diagnosing workflow processes using Woflan. H.M.W. Verbeek, T. Basten, W.M.P. van derAalst. Computer J. 44(4): 246-279 (2001)

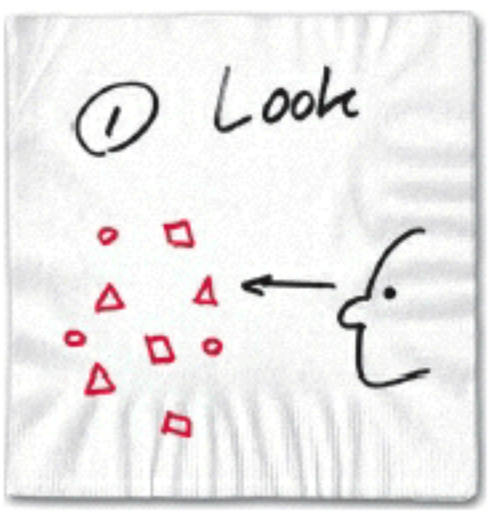
http://www.padsweb.rwth-aachen.de/wvdaalst/publications/p110.pdf

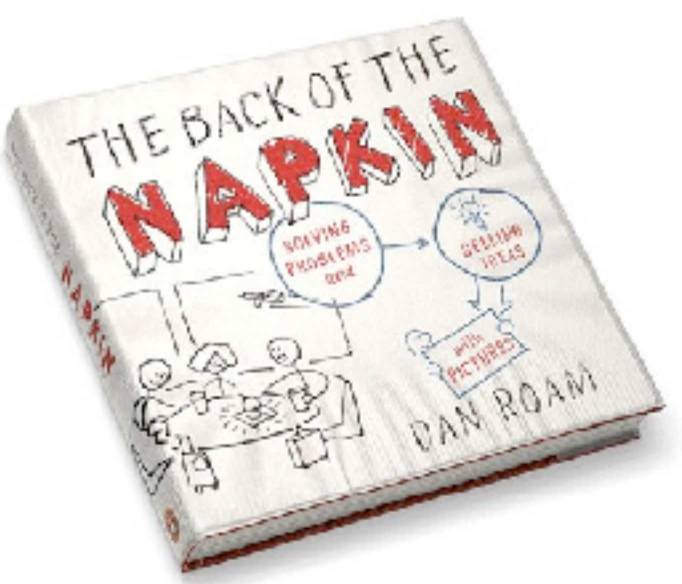
YAWL

https://yawlfoundation.github.io

Modelling

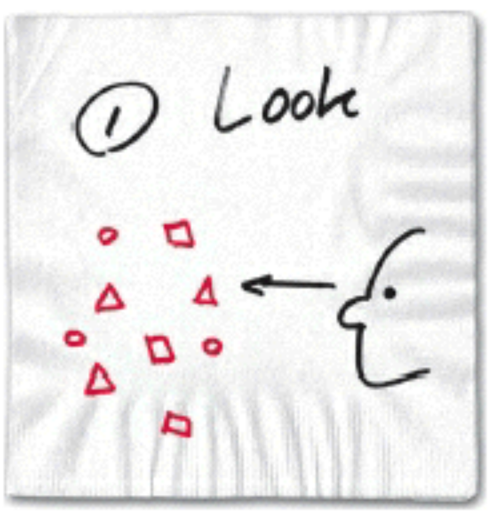
Giving shape to ideas, organizations, processes, collaborations, practices

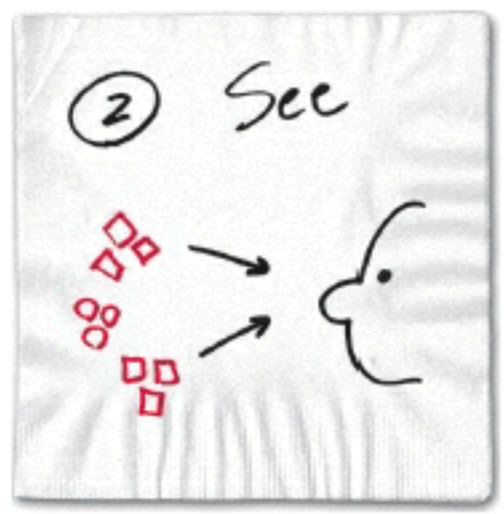


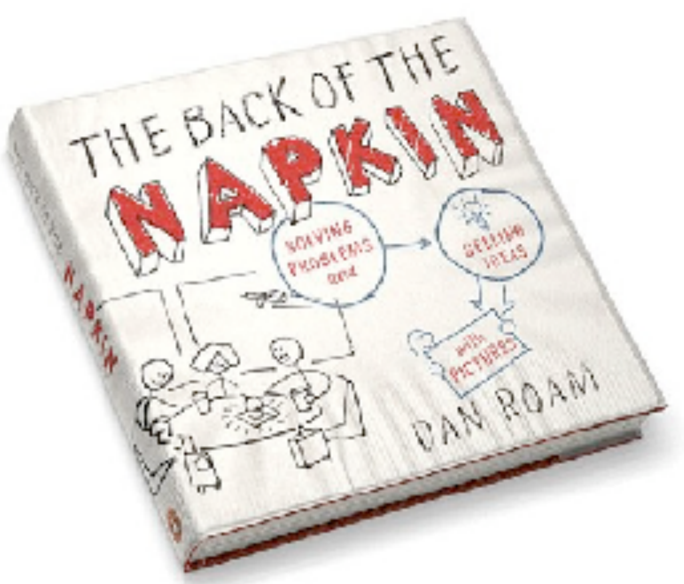


Modelling

Giving shape to ideas, organizations, processes, collaborations, practices

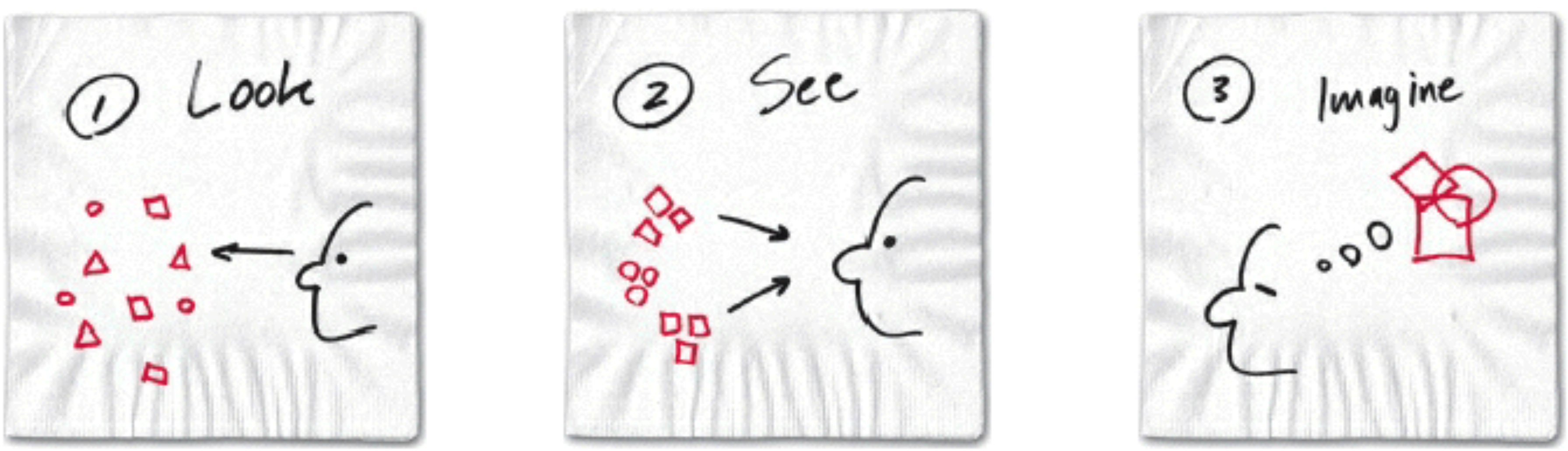


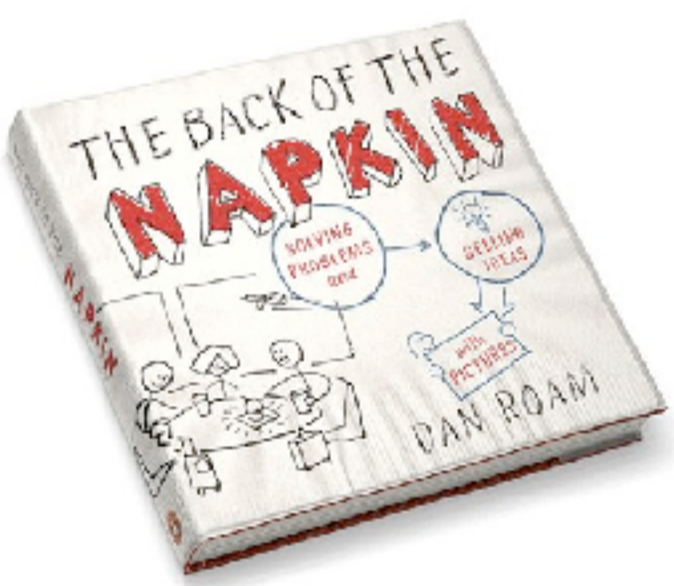




Modelling

Giving shape to ideas, organizations, processes, collaborations, practices

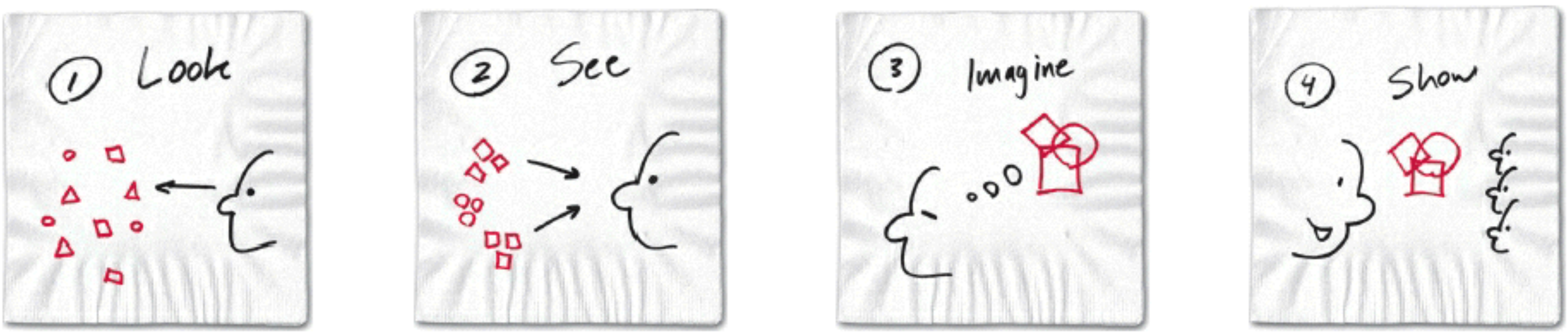


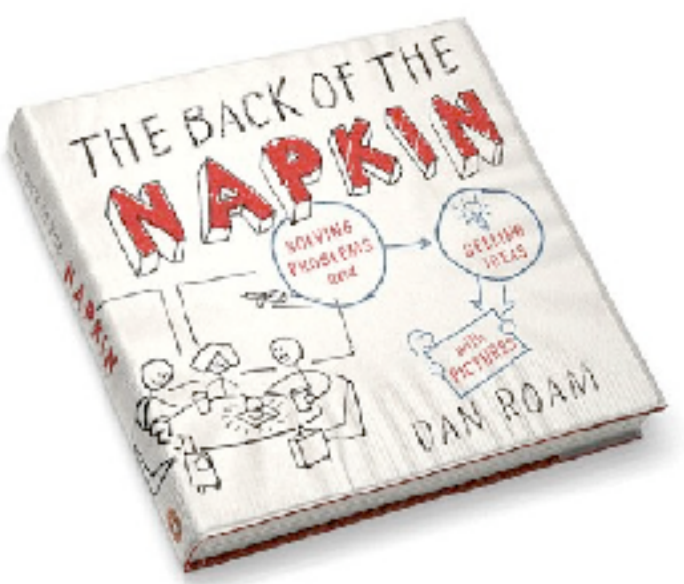


To analyse them

Modelling

Giving shape to ideas, organizations, processes, collaborations, practices





To analyse them

To communicate them to others

To change them if needed

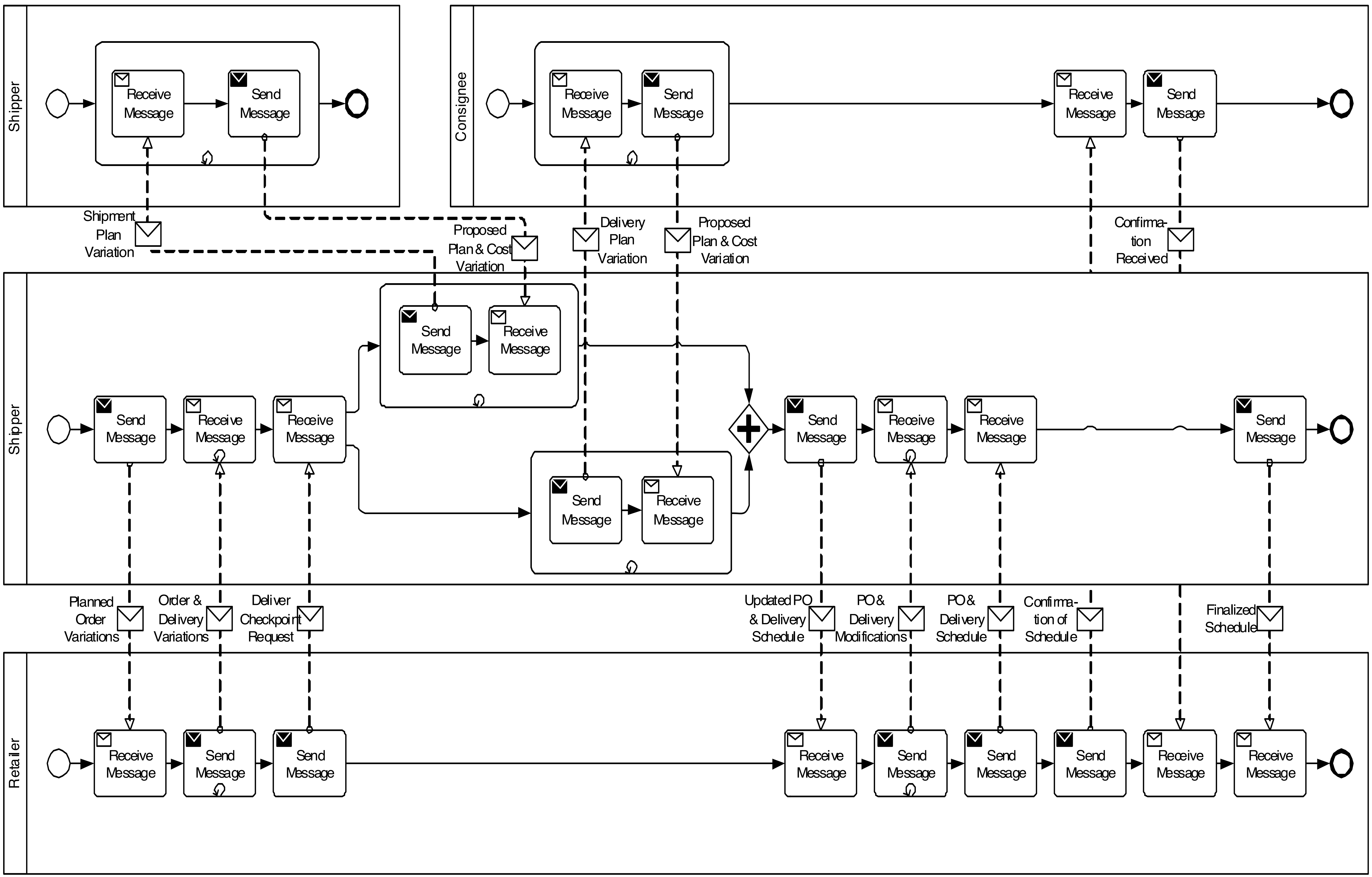
Quoting Michelangelo

Every block of stone has a statue inside it and it is the task of the sculptor to discover it

Quoting Michelangelo

Every organization has some processes running inside it and it is the task of the designer to discover them

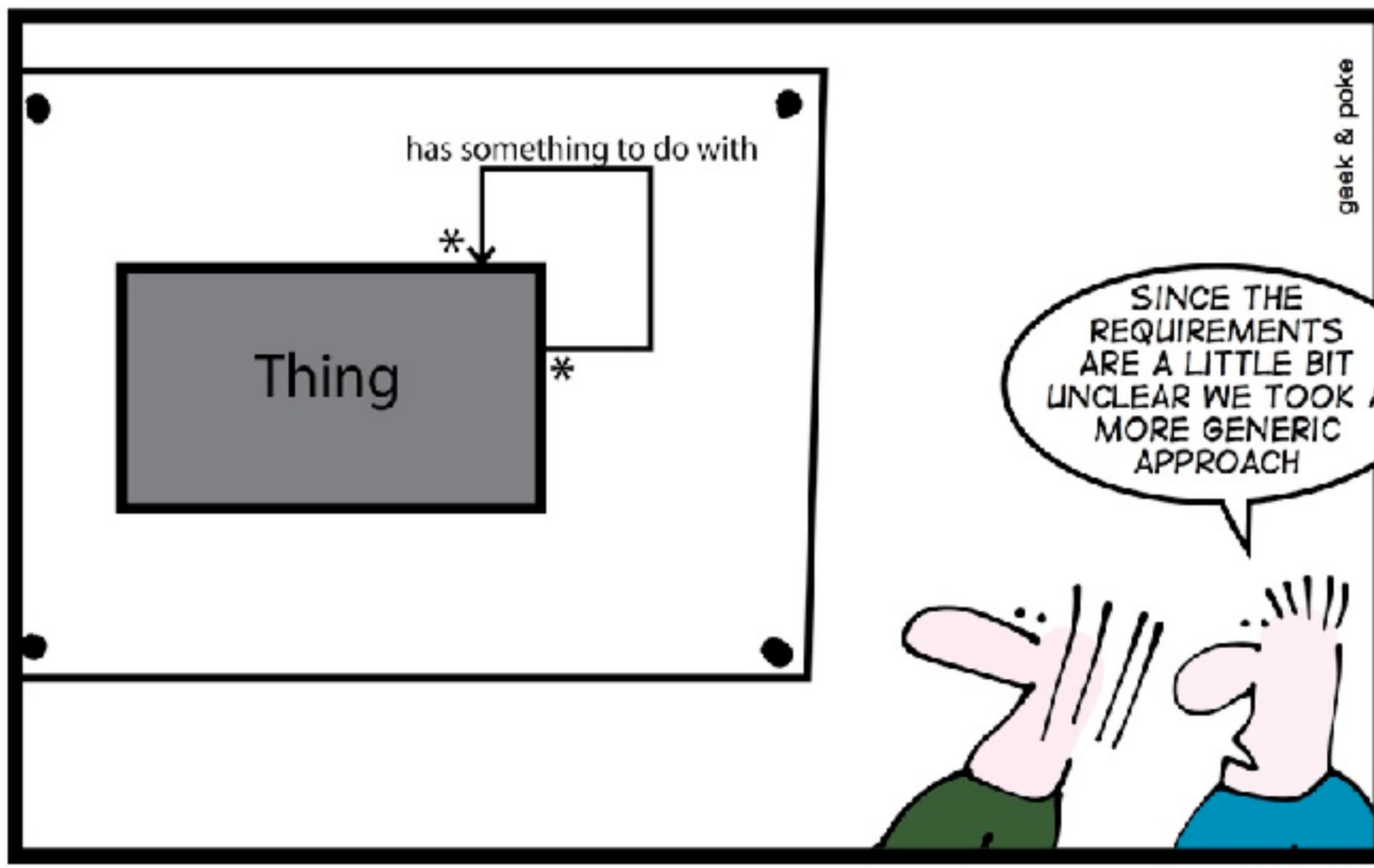
A taste of BPMN



What

Data and processes

Traditionally, information systems
used **information modelling** as a starting point

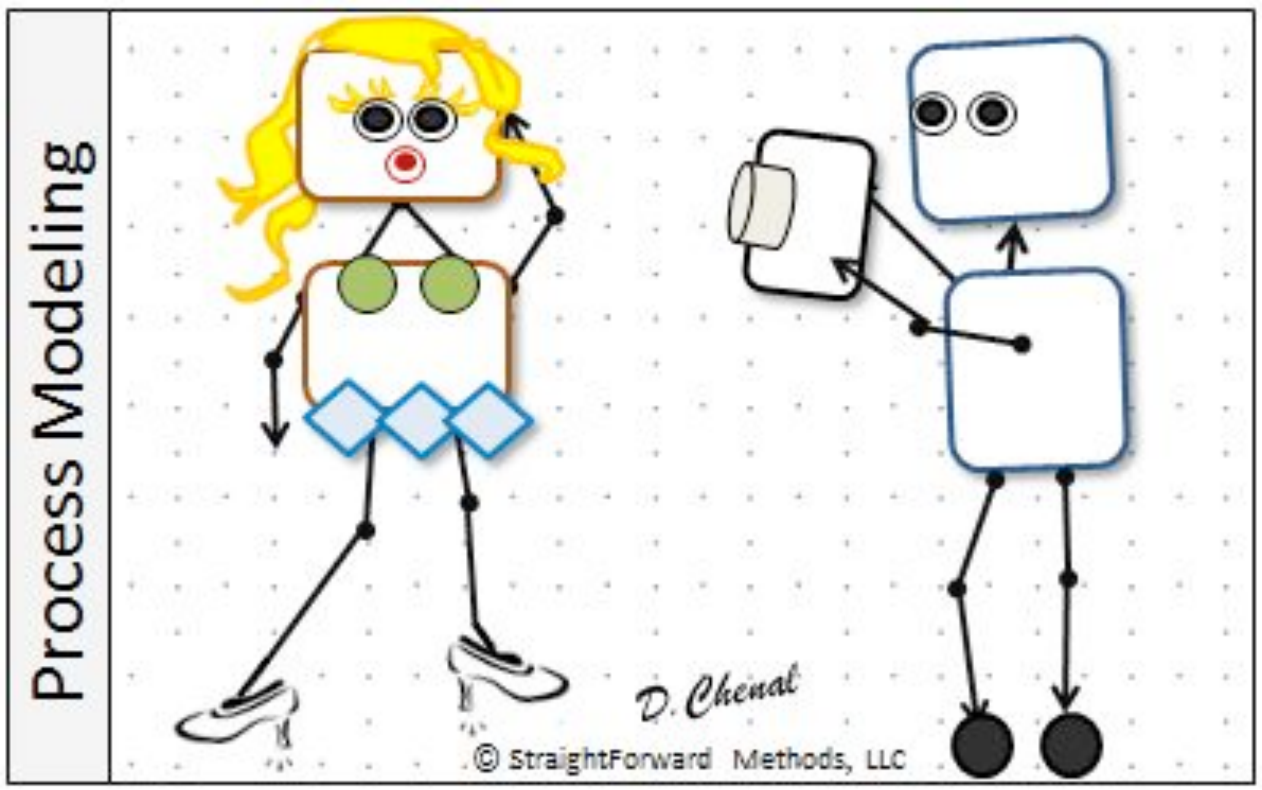


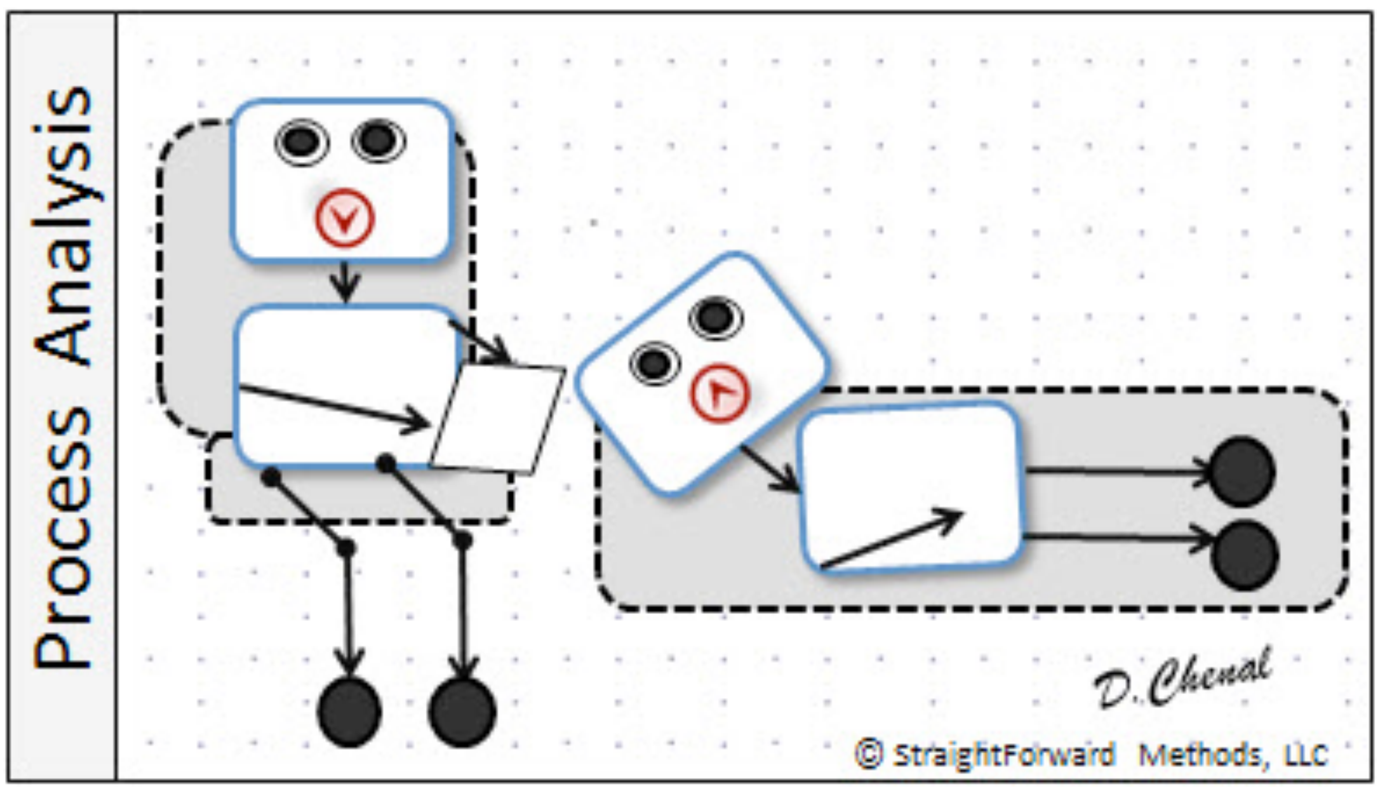
HOW TO CREATE A STABLE DATA MODEL

How

Data and processes

Nowadays, **processes** are of equal importance and need to be supported in a systematic manner

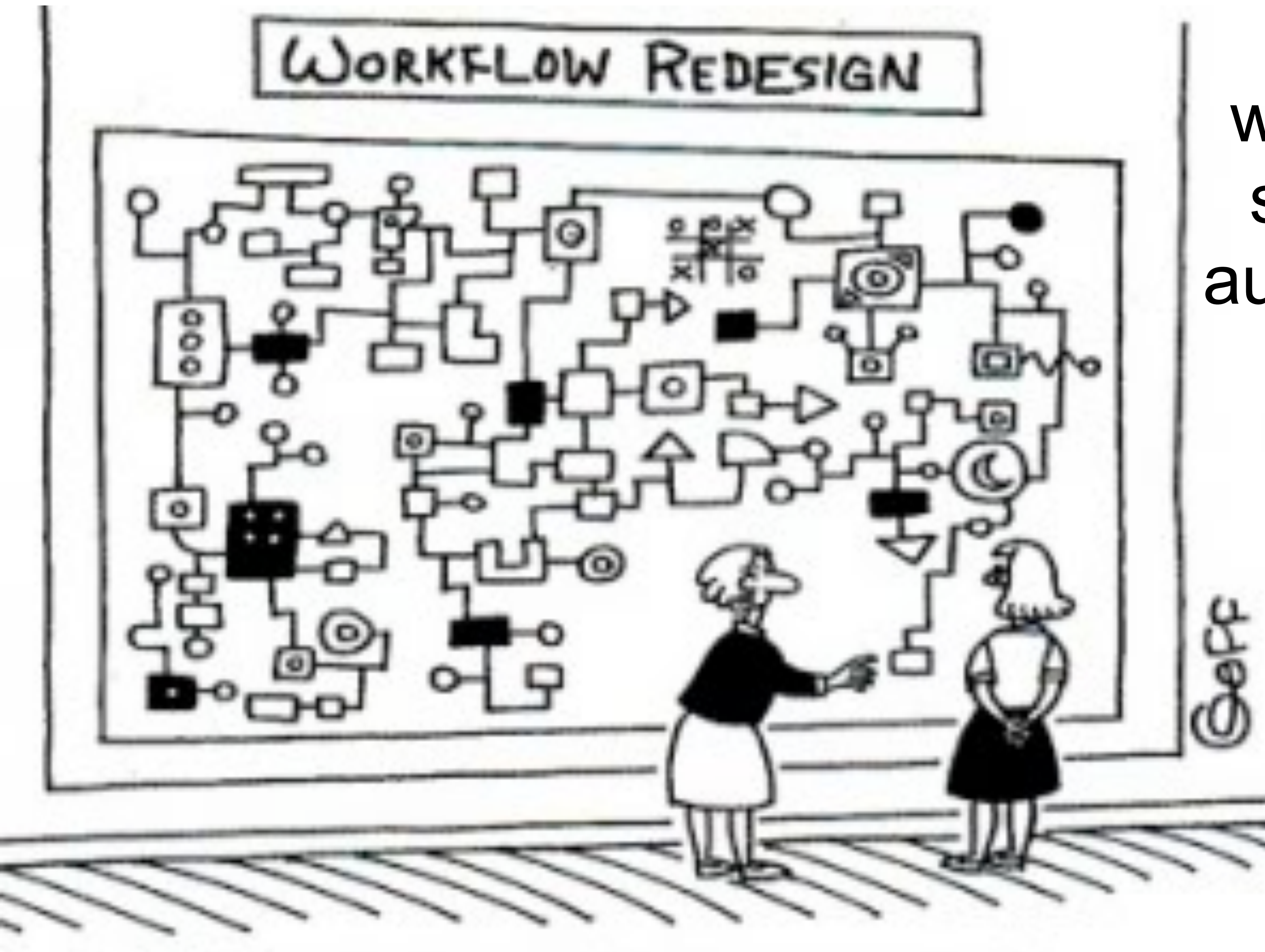




Motivation

- Each product is the outcome of a number of activities performed
- Because of modern communication facilities:
  - traditional product cycles not suitable for today's dynamic market
- Competitive advantages of successful companies:
  - the ability to bring new products to the market rapidly and
  - the ability to adapt an existing product at low cost
- Business processes are the key instrument:
  - to organize these activities
  - to improve the understanding of their relationships
- IT is an essential support for this aim

Workflow wave



"And this is where our ED workflow redesign team went insane."

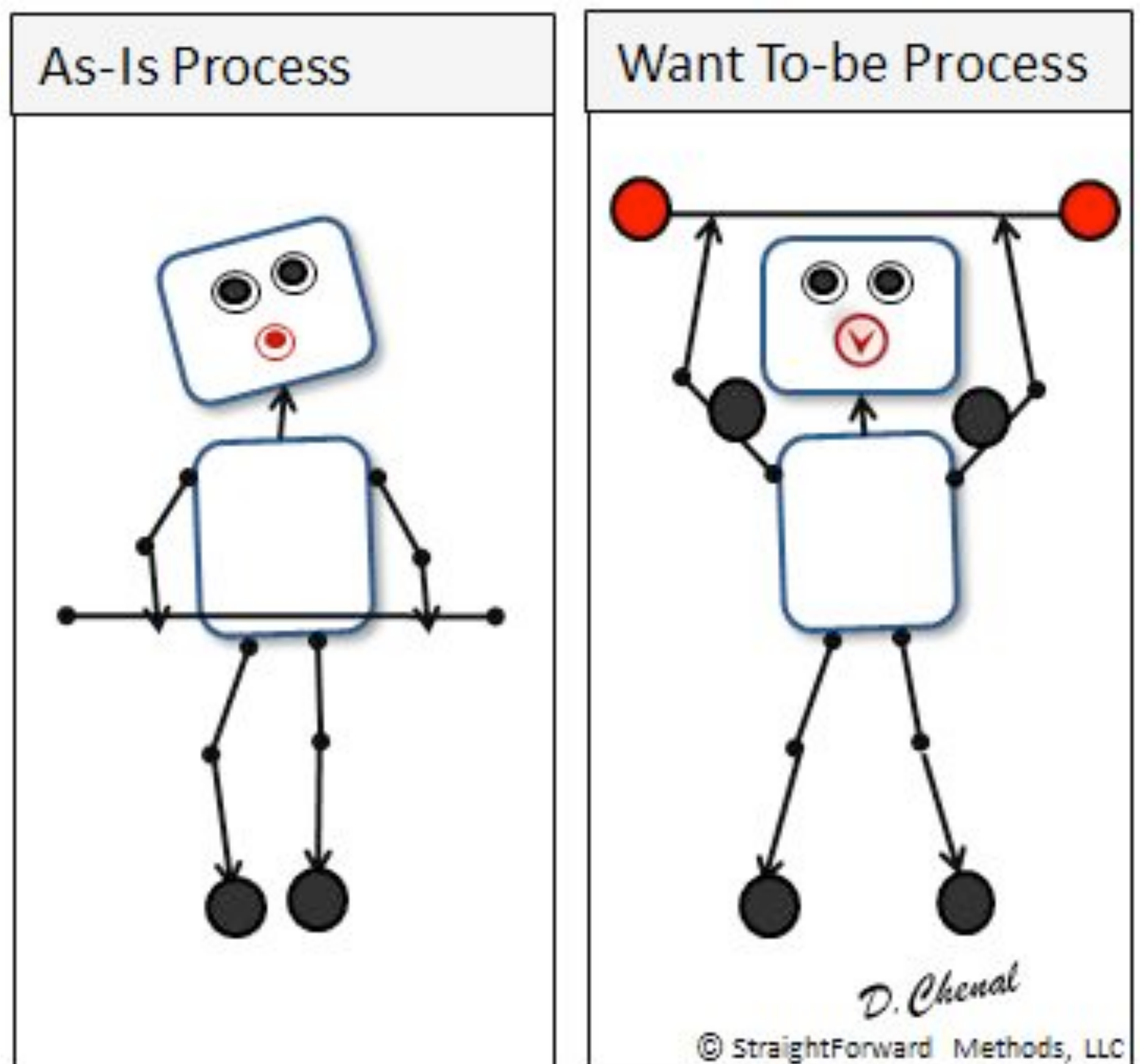
In the mid-nineties, workflow management systems aimed to the automation of structured processes

but their application was restricted to only a few application domains

Process awareness

BPM moves from workflow management systems (intra-organization)

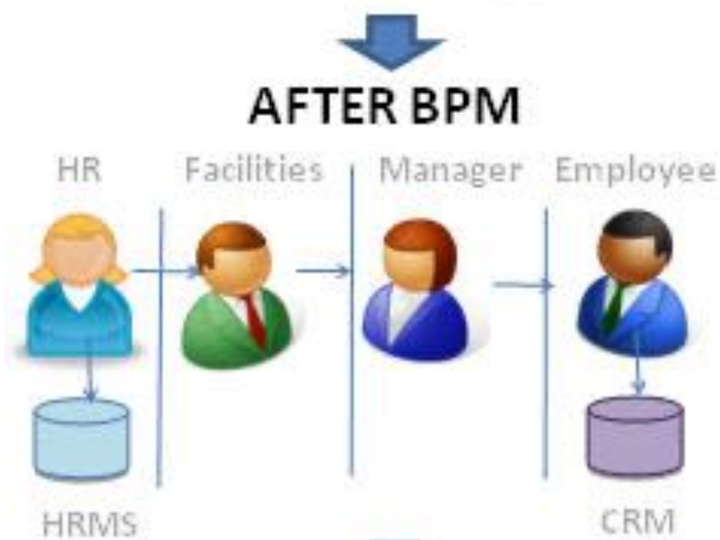
to the broader perspective of process-aware information systems (inter-organizations)



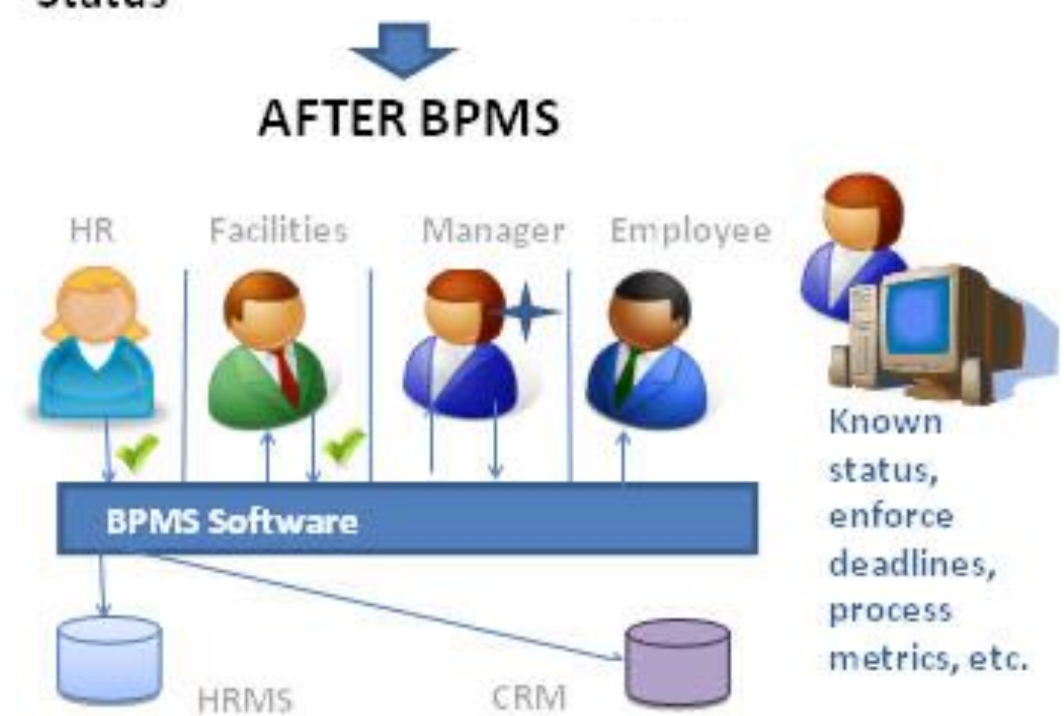
Process awareness

Know end goal

Handled differently every time



Know how to handle each occurrence



Benefits of BPM

automate workflow and orchestrate processes, reduce risk of errors, provide metrics, real time status, enforce deadlines, validate data, reduce training costs, …

What is the BPM maturity of your organization?

5 excellence

BPM is implemented enterprise-wide.
A continuous review & improvement process is implemented to exchange lessons-learned & address required changes proactively.

4 managed

BPM is implemented.
(People assigned. Communication to relevant people done. Training done. etc.)

3 defined

BPM is defined.

Implementation is yet missing or ongoing.

2 awareness

Awareness of BPM exists in the organization.

(Planning) activities have started for the definition of the subject.

1 initial

No structured BPM activities in the area of responsibility of the stakeholder.

Why BPM?

Highly relevant for pratitioners

Offers intellectual challenges for software developers and computer scientists

BPM angles

Analysis: simulation,
verification, process mining, ...

Influences: business aspects,
social aspects, training, education, ...

Technologies: interoperability,
standardization efforts,
service orientation, ...

Essential concepts

Different educational backgrounds and interests are in place

This course is not about a particular XML syntax (e.g., BPMN) or tool (e.g. ProM)

It is about using some process languages to describe, single out, relate, compare essential concepts

Which target?

Formal methods people

-investigate properties
-detect and correct deficiencies
-abstract from "real world"

Software develop people

-provide robust and scalable sw
-integration of existing sw
-look at new technology trends

Business admin people

-increase customer satisfaction
-reducing costs
-establishing new products

BPM aims at

robust and correct realization of business processes in software that increases customer satisfaction and ultimately contributes to the competitive advantage of an enterprise

Abstraction

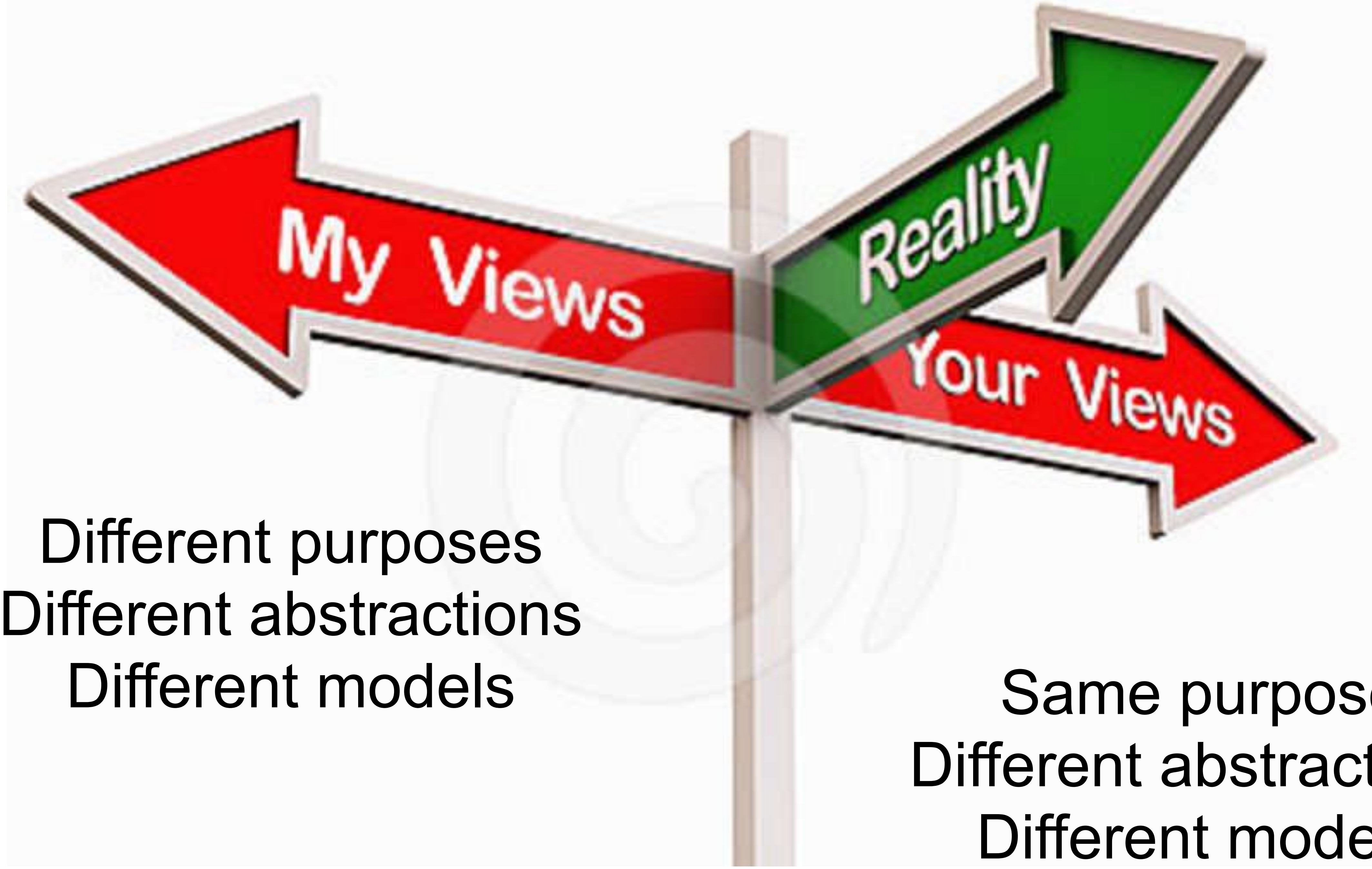
- Business admin people
  - IT as a subordinate aspect (for expert technicians)
  - This course: too much math!
- Software develop people
  - Current technology trend as main concern
  - This course: too abstract!
- Formal methods people
  - Underestimate business goals and regulations
  - This course: too much handwaving!

Abstraction as the key to achieve some common understanding, to build a bridge between views...

Levels of abstractions

One object,
many views

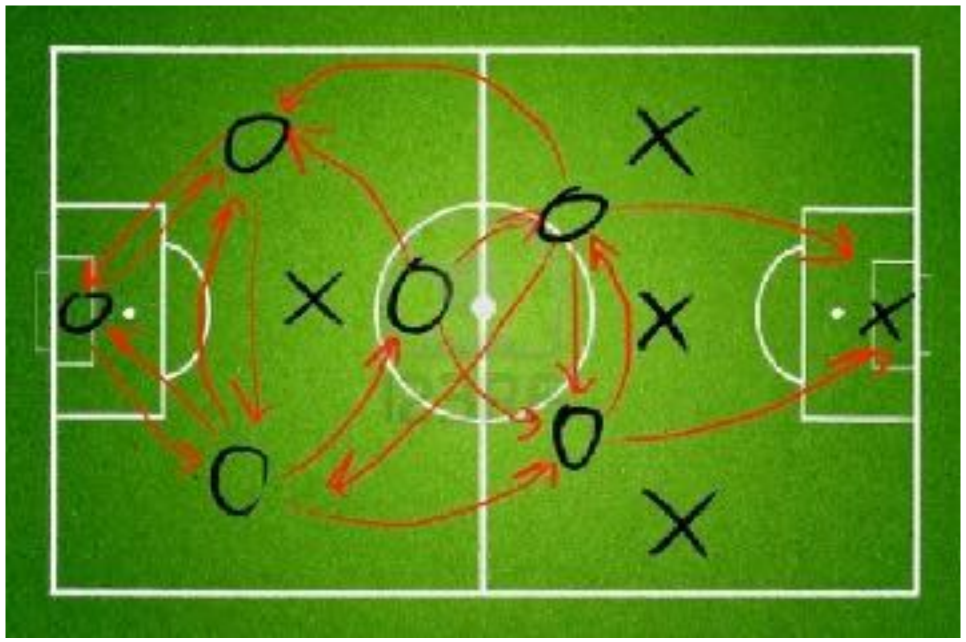
Different views are common

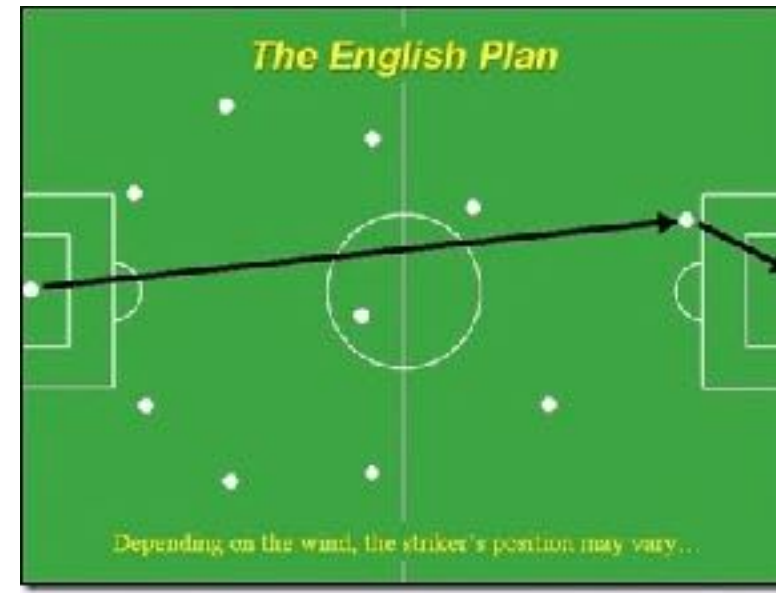


Different purposes
Different abstractions
Different models

Same purpose
Different abstractions
Different models

Everybody want~~s~~ed to be the Italian soccer team coach





What about the adversaries?

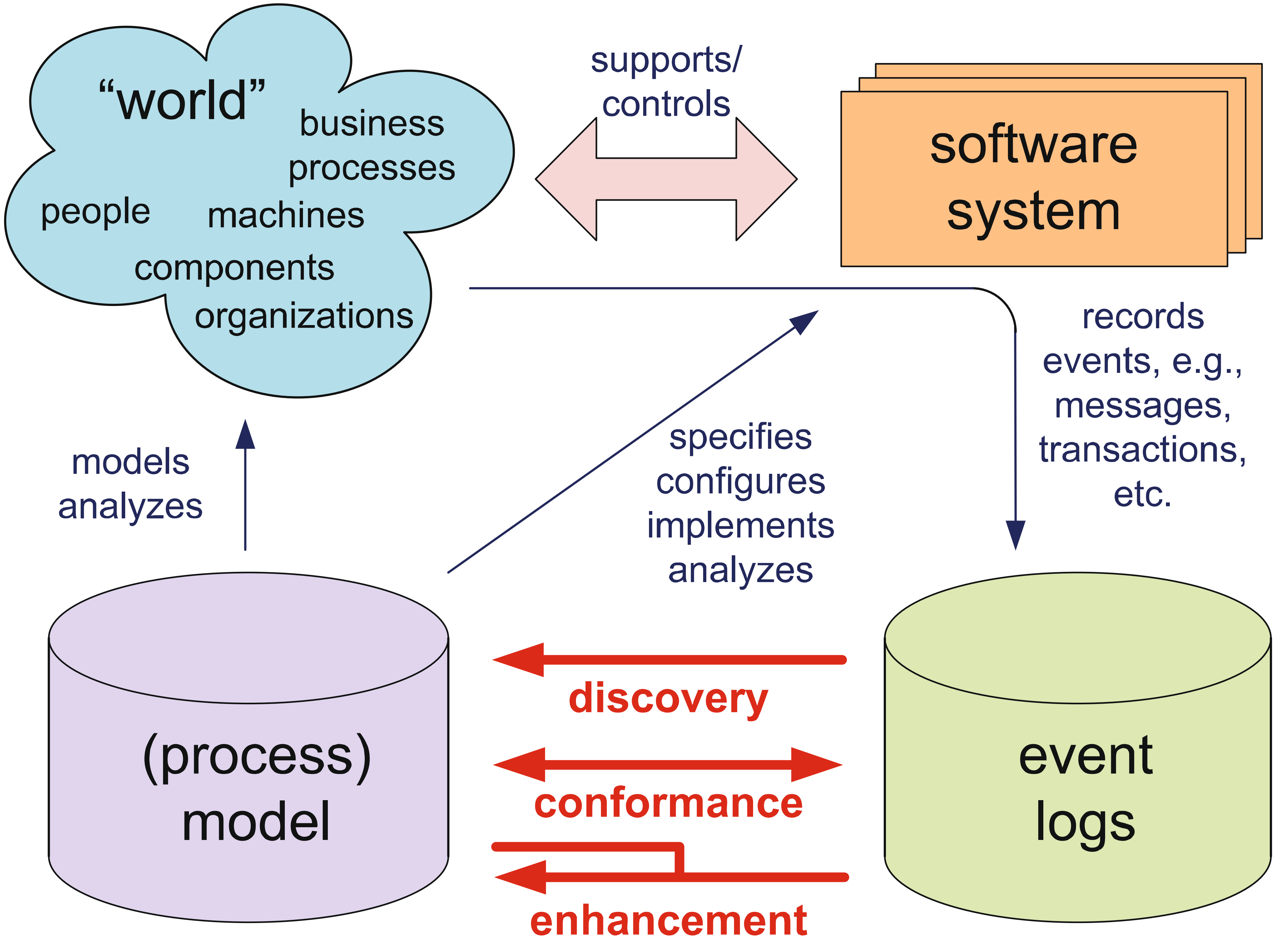
Can we find out their plan?

Knowing it would be quite helpful

Any idea how to?

(abstractions can be designed but can also be derived)

A taste of Process Mining



Digression:
formal modelling
and the importance of proofs

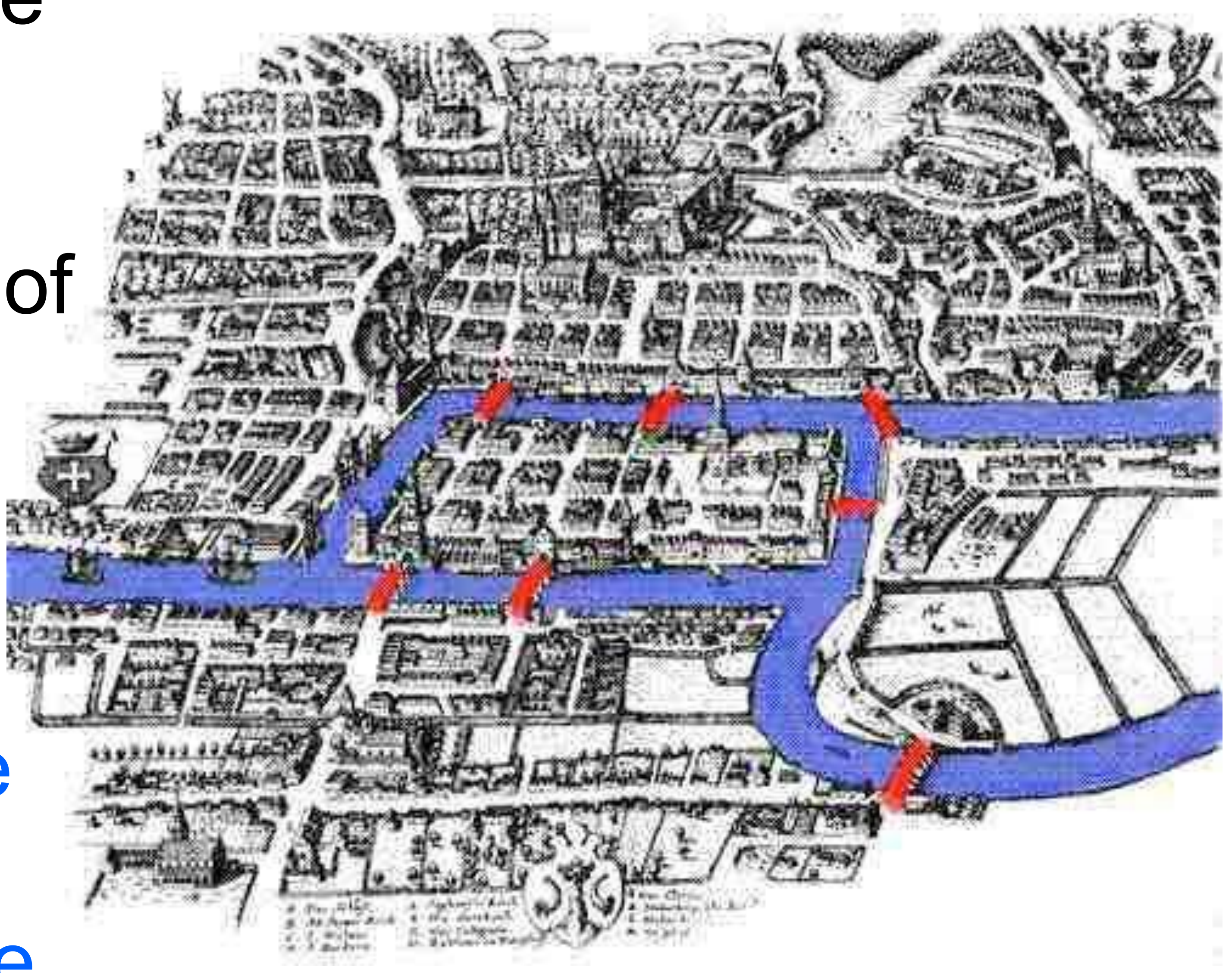
Digression...

Let us travel back in time to the 18th century, in the beautiful Prussian city of Königsberg:

it was thanks to the layout of that city that Leonhard Euler answered a puzzle which eventually contributed to the birth of two new areas of maths known as topology and graph theory.

Digression…

Königsberg was built on the bank of the river Pregel. Seven bridges connected two islands and the banks of the river (see map).

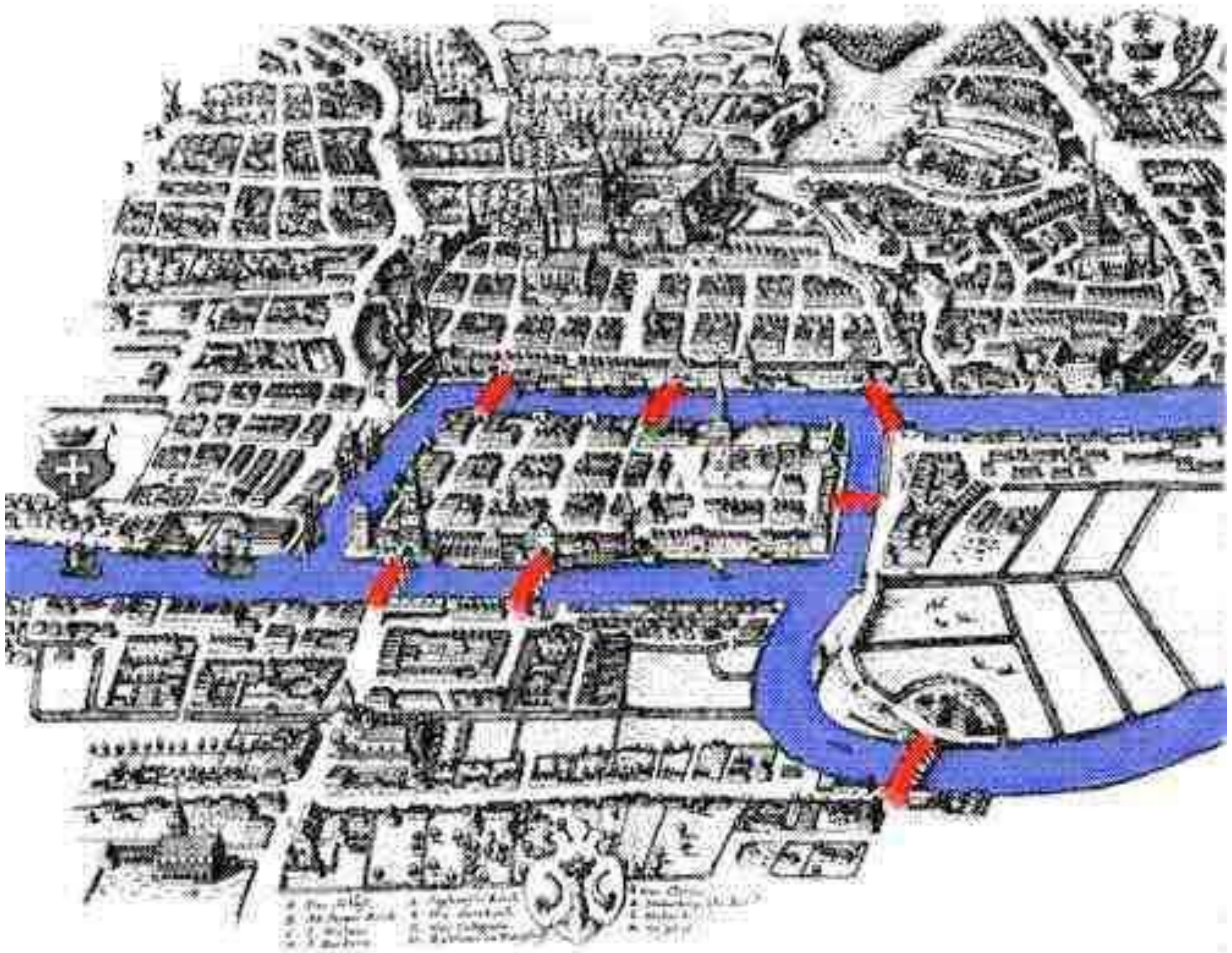


A popular pastime of the residents was to try to cross all the bridges in one complete circuit (without crossing any of the bridges more than once).

Digression...

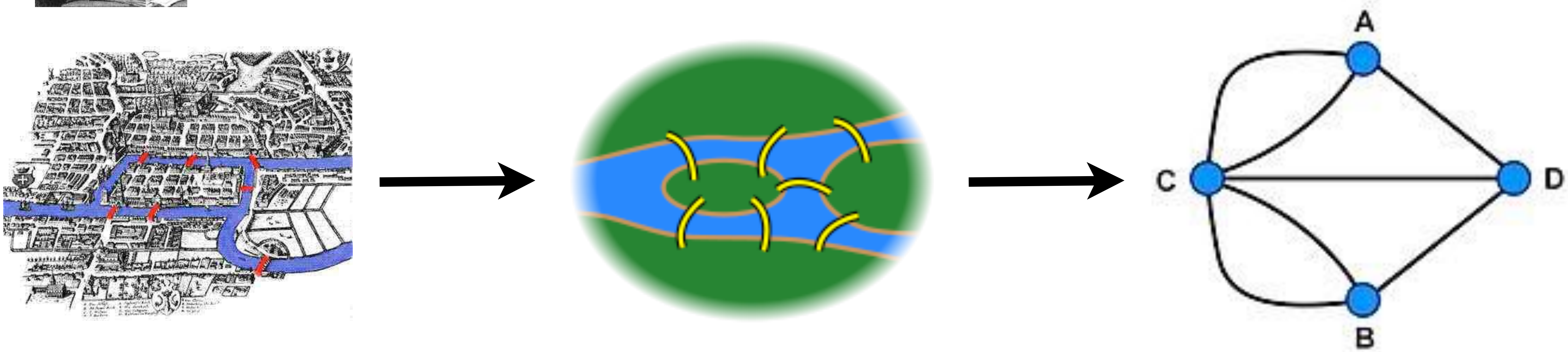
A seemingly simple task was more than tricky...

Nobody had been able to find a solution to the puzzle when Euler first heard of it and, intrigued by this, he set about **proving** that no solution was possible!



Digression...

In 1736, Euler analysed the problem by converting the map into a more abstract diagram… and then into a graph (a formal model):



areas of land separated by the river were turned into points, which he labelled with capital letters.
Modern graph theorists call these vertices or nodes.

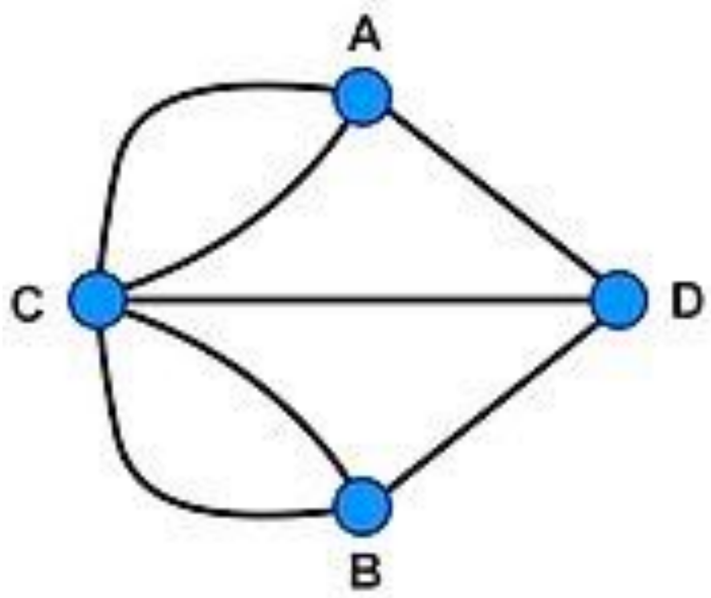
The bridges became edges between nodes.

Digression...

Modeling activities require several steps of abstraction that must preserve the set of solutions: in other words the abstractions must preserve the topology of the problem.

Original problem: *seven bridges of Königsberg*

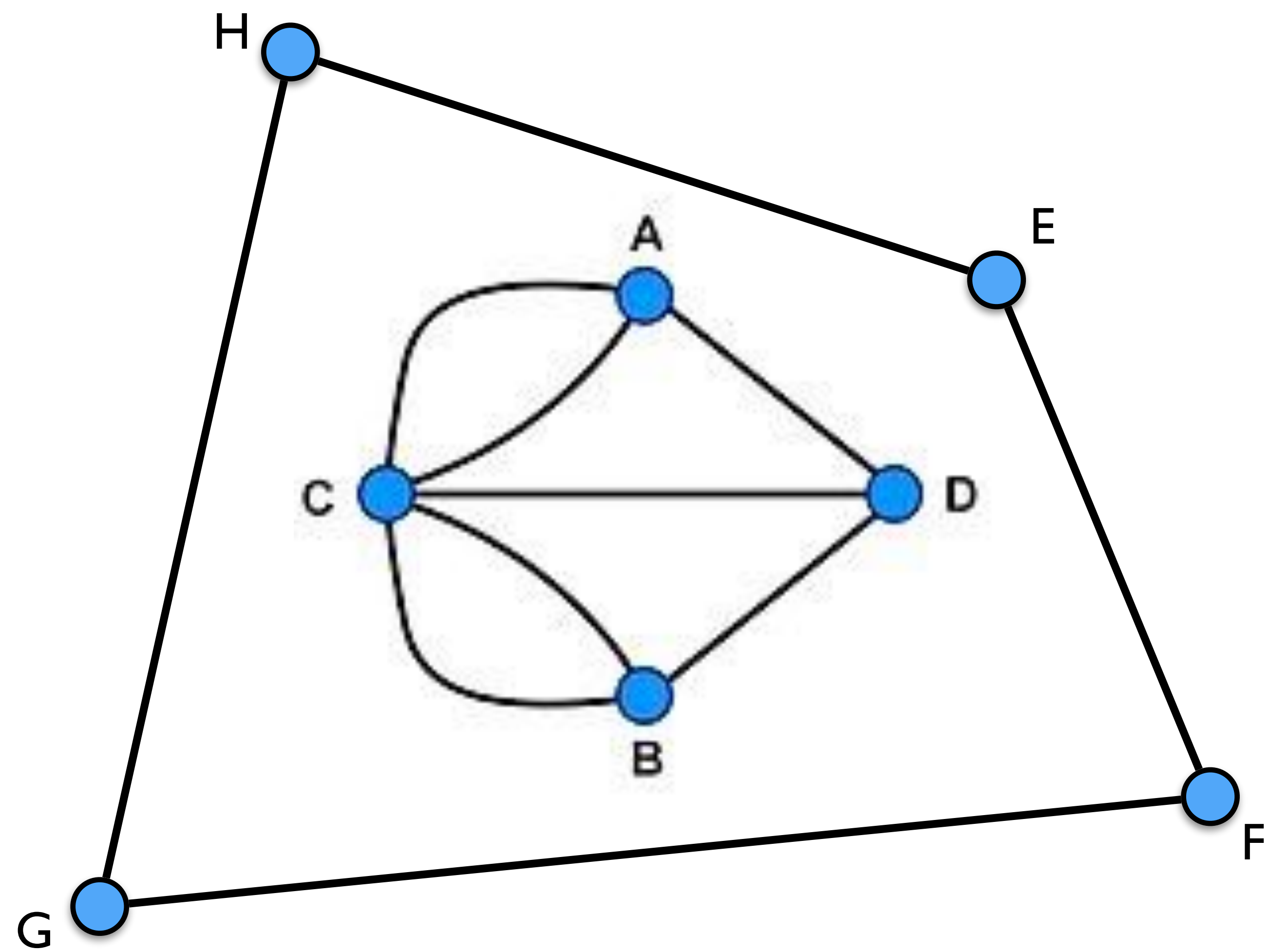
Graph problem: *redrawing this picture without retracing any line and without picking your pencil up off the paper*



Generalized problem: *given a connected graph, find a circuit that visits every edge precisely once, if it exists*

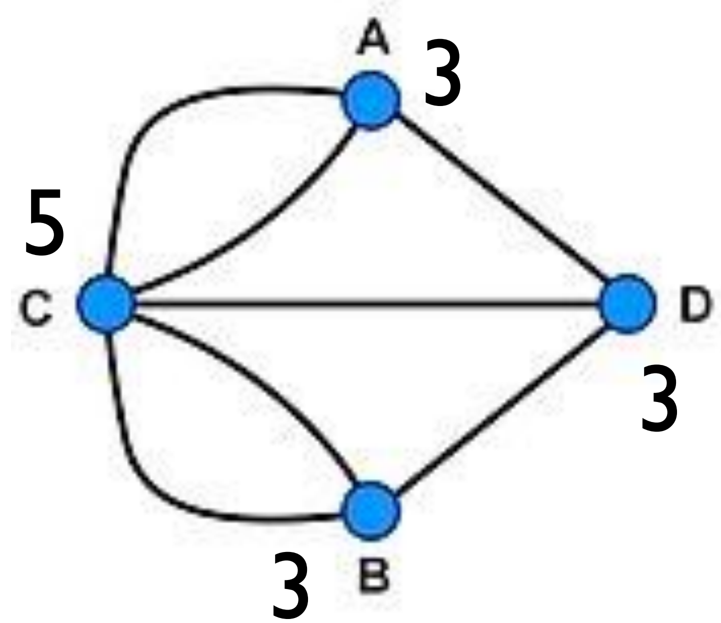
Digression...

A non-connected graph



Digression...

Informal reasoning:
All the vertices in the picture are connected to an **odd** number of edges

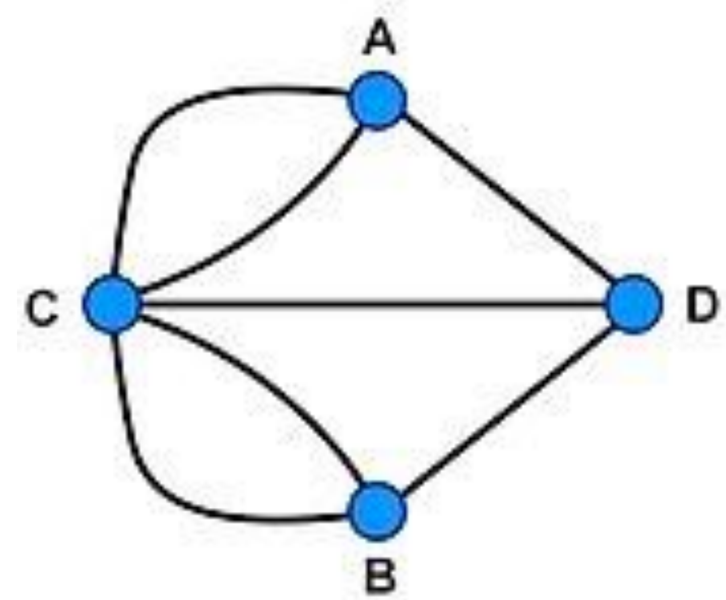


In a circuit every time you enter a node you must be able to leave it: thus, every vertice must be connected to an **even** number of edges!

This suffices to establish that no solution can be found!

Digression...

Take one of these vertices, say A, and start trying to trace the figure out without picking up your pencil: then two edges are left from/to A.



Next time you arrive in A, one edge will be left, and when you will leave A, no edge to re-enter it will be left!

Analogously for B, C, D: No circuit is possible!

Digression...

Formal reasoning:

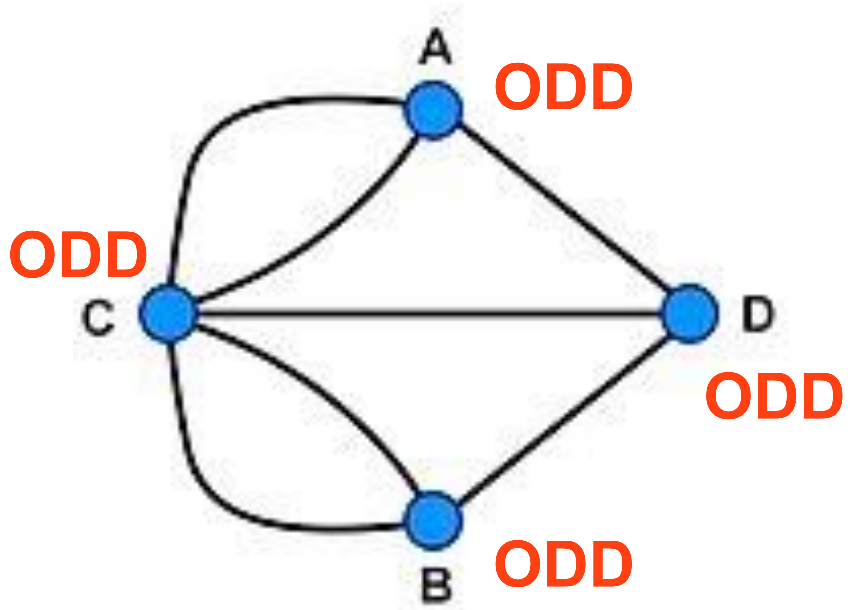
Definition: A path is a (finite) sequence of contiguous edges.
It is a circuit if it ends in the same vertex where it starts.

Definition: An Eulerian path is a path that passes through every edge of the graph once and only once.

Definition: The number of edges attached to v is called degree of v.
A vertex is called odd if it has an odd degree, even otherwise.

Theorem: A (connected) graph G contains an Eulerian circuit if and only if each vertex is even.

Digression...



Theorem: A (connected) graph G contains an Eulerian circuit if and only if each vertex is even.

Digression...

Proof of necessity:
(If G has an Eulerian circuit, then any vertex is even)

Suppose G contains an Eulerian circuit C.

Then, for any choice of vertex v, C contains all the edges that are adjacent to v.

Furthermore, as we traverse along C, we must enter and leave v the same number of times, and it follows that v must be even. **qed**

While this proof of necessity was given by Euler, the converse was not stated in his paper.

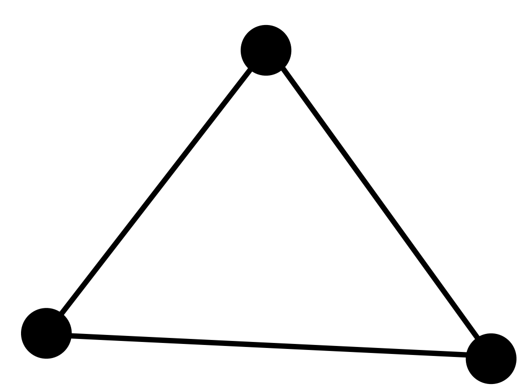
It is not until 1873 (137 years later) when a young German mathematician, Carl Hierholzer published the proof of sufficiency.

Digression...

Proof of sufficiency:
(If any vertex is even, then G has an Eulerian circuit)

The proof is by induction on the numbers of edges, ctd)

<u>Base case</u>: the smallest possible number of edges is 3 (i.e. a triangle) and the graph trivially contains an Eulerian circuit.



Digression...

Proof of sufficiency: (by induction on the numbers of edges, ctd)

Inductive case:
Inductive hypothesis: Let us assume that any connected graph H that contains k or less than k edges and such that every vertex of H is even, contains an Eulerian circuit.

Now, let G be a graph with k + 1 edges, and such that every vertex is even. We want to prove G has an Eulerian circuit

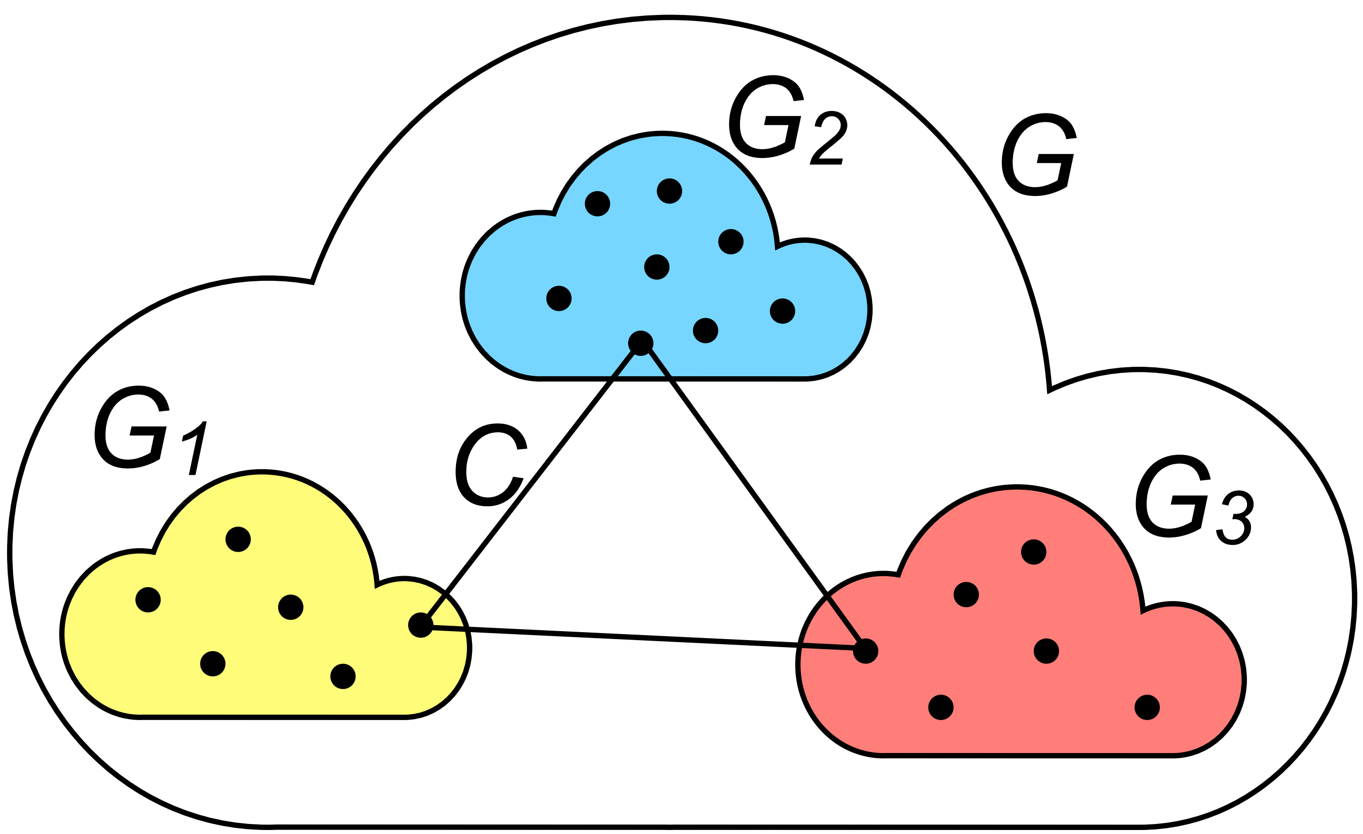
Since there is no odd vertex, G cannot be a tree (no leaves).
Thus, G must contain at least one cycle C.

...

Digression...

Proof of sufficiency: (by induction on the numbers of edges, ctd)
… Remove the edges of C from G, and take the remaining graph G'.

Since removing C from G may disconnect the graph, G' is a collection of connected components, namely G_1 *,* G_2 *, . . . , etc.*

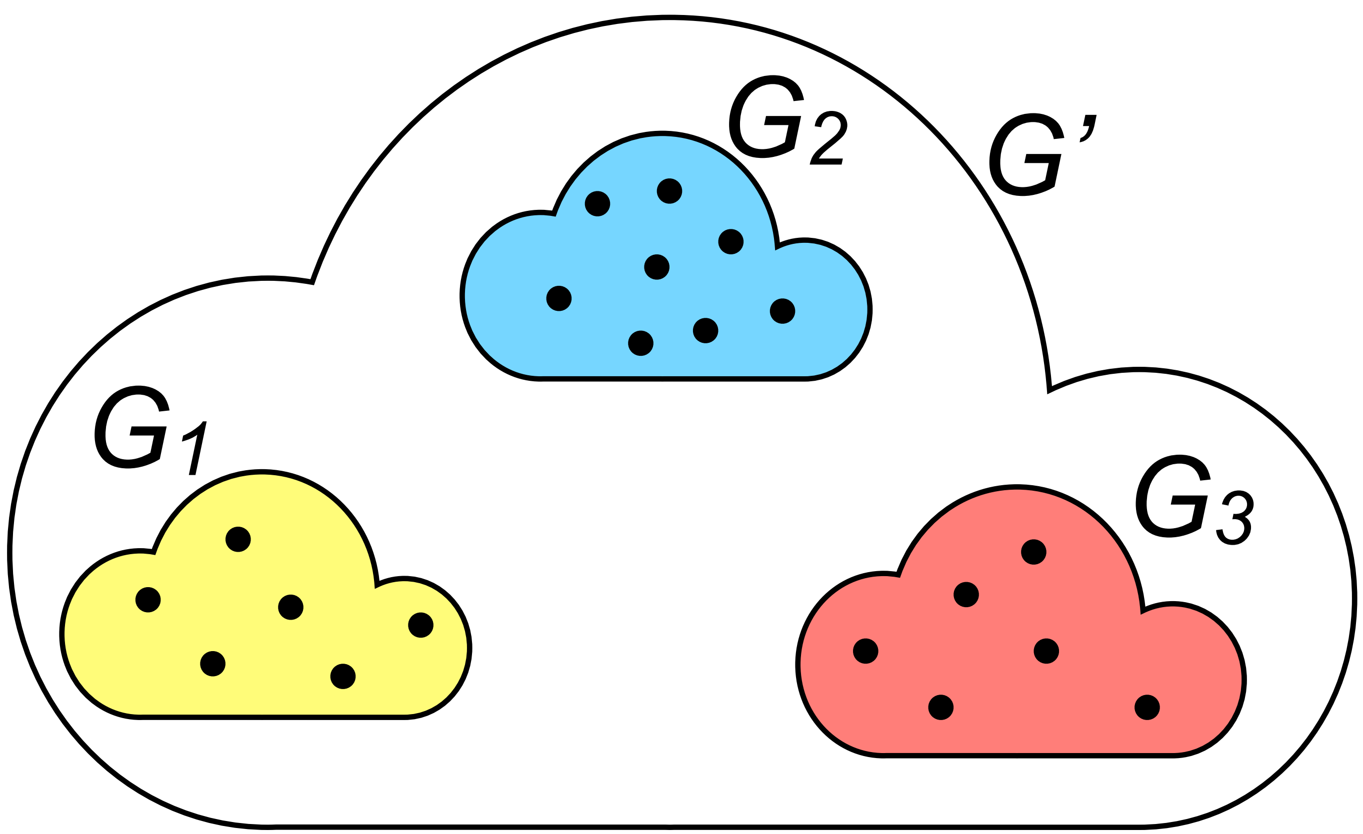


...

Digression...

Proof of sufficiency: (by induction on the numbers of edges, ctd)
… Remove the edges of C from G, and take the remaining graph G'.

Since removing C from G may disconnect the graph, G' is a collection of connected components, namely G_1 , G_2 , . . . , etc.



...

Digression…

Proof of sufficiency: (by induction on the numbers of edges, ctd)
… When the edges in C are removed, each vertex loses an even number of adjacent edges.
Thus, the parity of each vertex is unchanged in G'.

It follows that, for each connected component of G', every vertex is even (and G' must have less than k edges).

Therefore, by the inductive hypothesis, each of G_1 , G_2 , . . . *has its own Eulerian circuit, namely* C_1 , C_2 , *etc.*

…

Digression…

Proof of sufficiency: (by induction on the numbers of edges, ctd)
… We can now build an Eulerian circuit for the whole graph G.

Pick an arbitrary vertex v from C.

Traverse the edges along C until we reach a vertex v_i that belongs to one of the connected components G_i.

Then, traverse G_i along its Eulerian circuit C_i until we traverse all the edges of C_i.

We are now back at v_i, and so we can continue on along C.

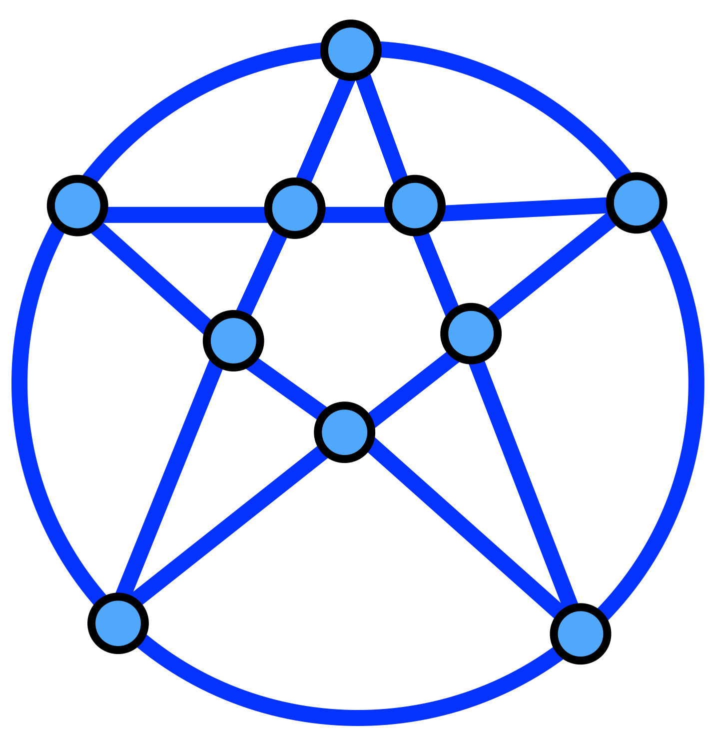
In the end, we shall return back to the first starting vertex v of C, after visiting every edge exactly once. **qed**

Digression...

The theorem, as such, is only an existential statement.

If the necessary and sufficient condition is satisfied, we'd like to find an Eulerian circuit.

The sufficiency proof gives us an algorithm to build Eulerian circuits: **recursively find a cycle, and then remove the edges of the cycle.**

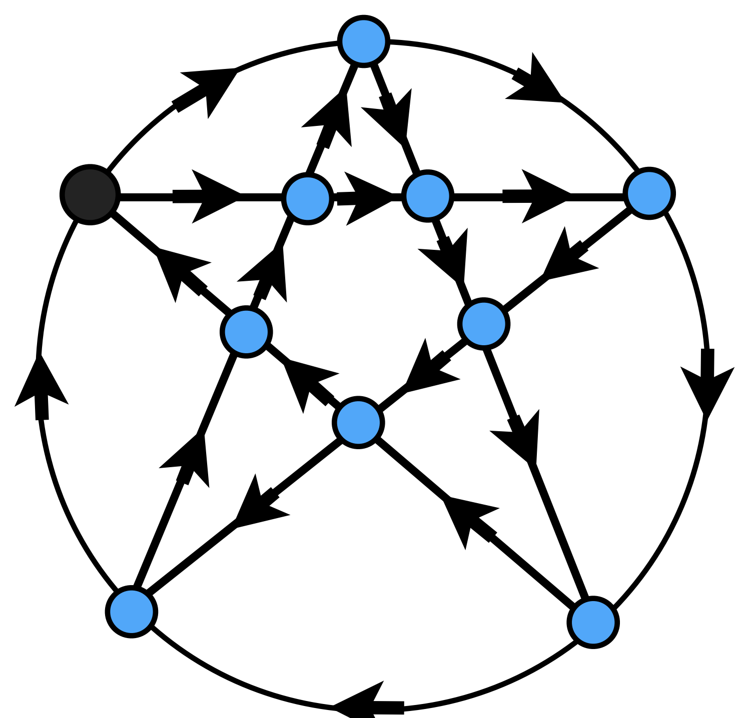


Digression...

The theorem, as such, is only an existential statement.

If the necessary and sufficient condition is satisfied, we wish to find an Eulerian circuit.

The proof naturally gives an algorithm to construct Eulerian circuits: **recursively find a cycle, and then remove the edges of the cycle.**



Digression...

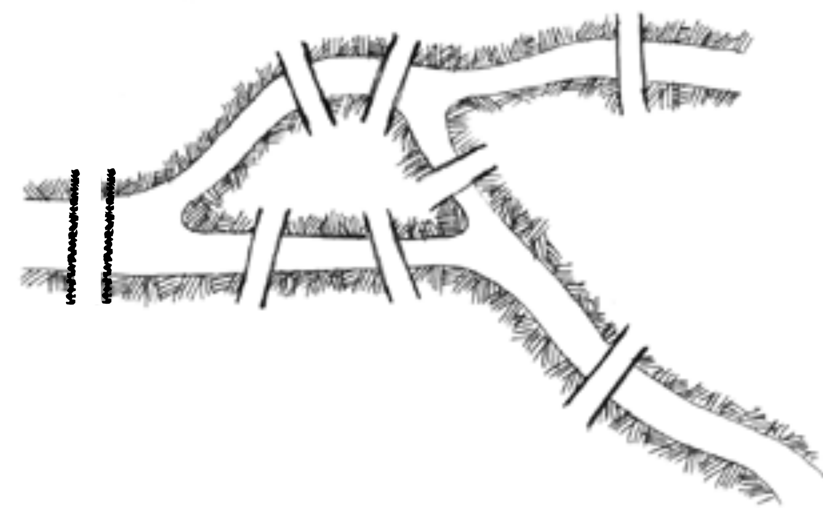
Theorem: A graph contains an **Eulerian path** if and only if there are 0 or 2 odd vertices.

Proof. (first part: if G has an Eulerian path then there are 0 or 2 odd vertices)
Suppose a graph G contains an Eulerian path P.
Then, for every vertex v, P must enter and leave v the same number of times, except when it is either the starting vertex or the final vertex of P.
When the starting and final vertices are distinct, there are precisely 2 odd vertices.
When these two vertices coincide, there is no odd vertex.

(second part: if G has 0 or 2 odd vertices, then there is an Eulerian path)
Conversely, suppose G contains 2 odd vertices u and v.
(The case where G has no odd degree vertex is shown in the previous Theorem.)
Then, temporarily add a dummy edge (u, v) to G.
Now the modified graph contains no odd vertex.
By the previous Theorem, this graph contains an Eulerian circuit C that includes (u, v).
Remove (u, v) from C, and now we have an Eulerian path where u and v serve as initial and final vertices.

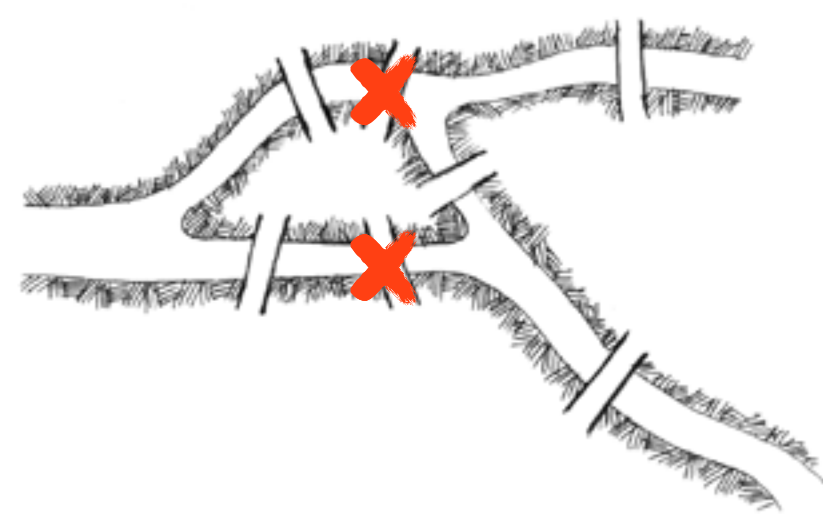
Digression...

In the late 19th century an eighth bridge was built (see map).
Was Königsberg Eulerised?



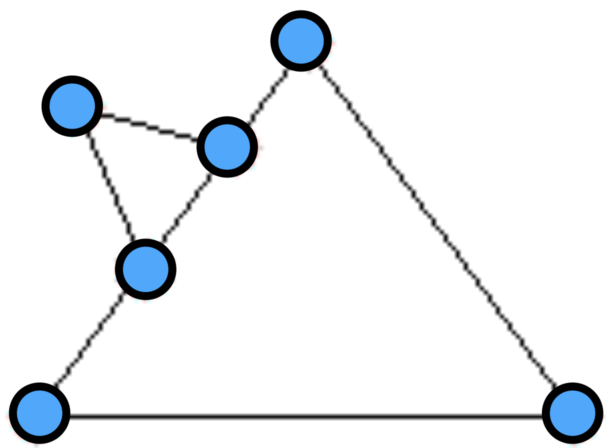
Exercise: prove that an Eulerian path can be found (but not a circuit)

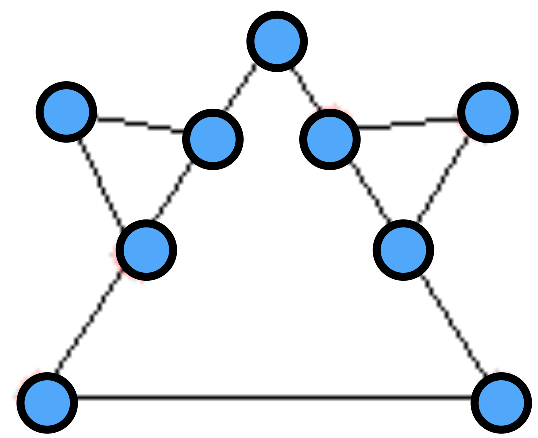
Sadly, in 1944 air raids obliterated most of the bridges. However, five bridges crossing were rebuilt (see map).
Was the city Eulerised?

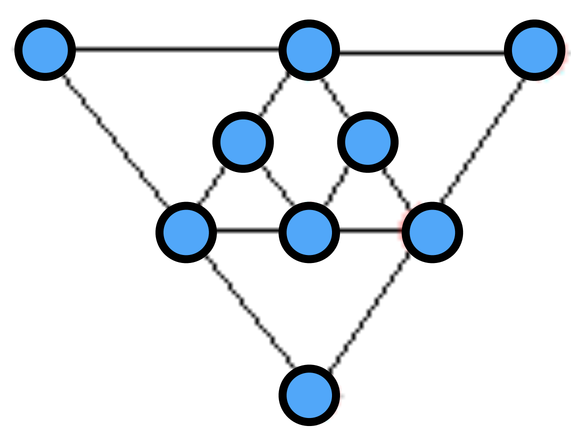


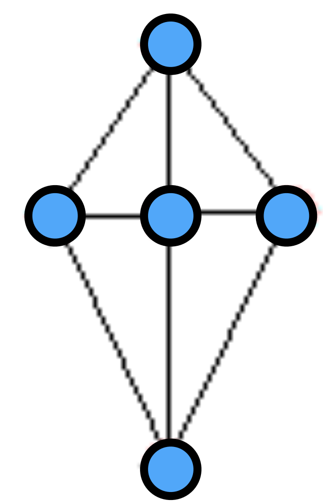
Exercise: prove that an Eulerian path can be found (but not a circuit)

Digression...









Exercises: find Eulerian paths/circuits in the graphs above or prove that they cannot exist.

Recap

Lessons learned

- Start with a concrete instance of the problem
- Abstraction: modeling and generalization, property preserving
- Visual notation: informal, intuitive
- Mathematical notation: rigorous, precise
- Theorems: alternative ways to find answers
- Proofs: construct solutions from formal reasoning
- Implementation: solve any concrete instance of the problem

Yet to learn

- Formal models used in prescriptive manner
- Correctness by design
- Separation of concerns
- Model discovery

Avoid bad designs!