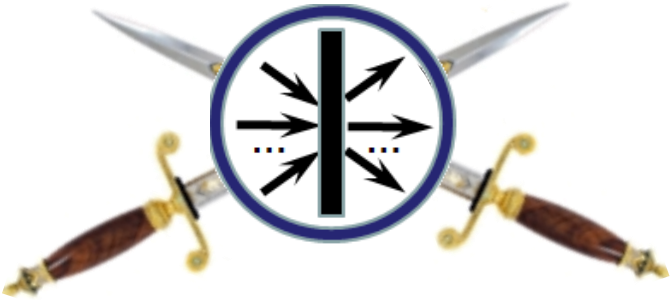


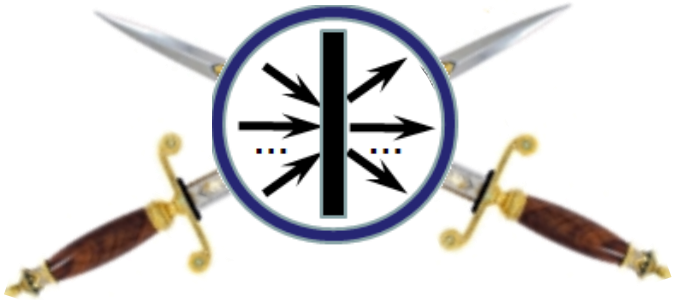
Admiral: Military Logistics Execution Monitoring Engine

Brief presentation of the concept behind the implementation

by **Vagan Terziyan**

Industrial Ontologies Group

The original idea of Admiral (September, 2011)



Admiral: Business Process Execution Monitoring Engine

This former version did not include an intelligent component as the new one

TiViT

The original idea has been appreciated

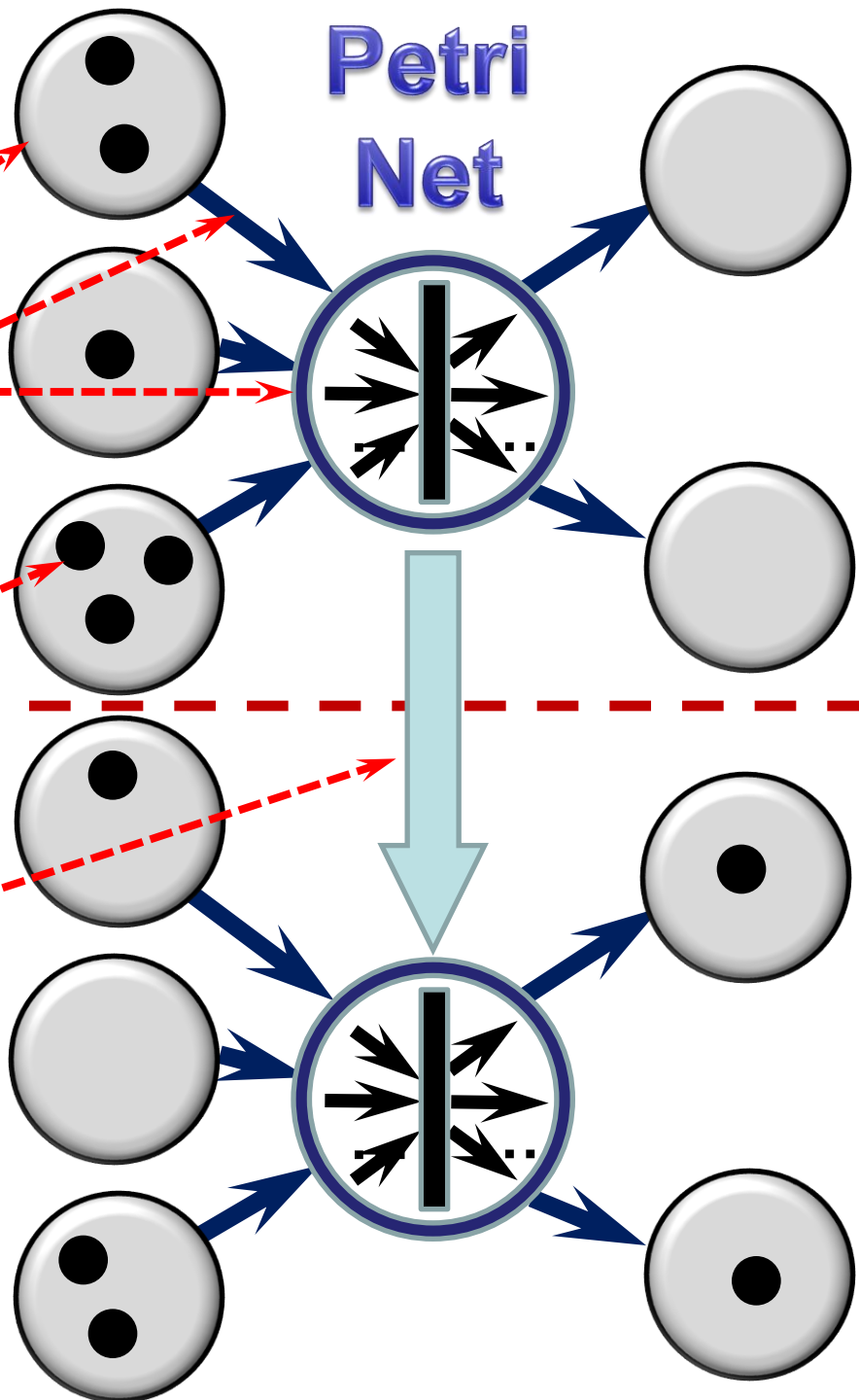
Industrial Ontologies Group

Reijo Paajanen
Tivit Oy:n toimitusjohtaja

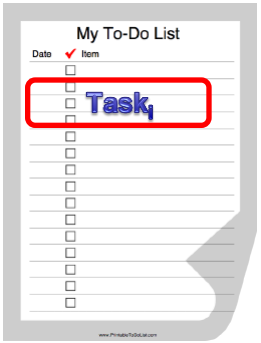
Vagan Terziyan
Head of IOG

"Vagan. This is a great proposal and actually the only practical way to proceed with Tivit because I am overloaded and Tivit has very limited resources. I am ready to do my best to comment specification and functionality and give my input in meetings. When we some day can have process engine up and running TIVIT is willing to consider the use of system and invest in some development steps as well as the work is reasonable..." Br Reijo. (From: Reijo.Paajanen@Tivit.fi, Mon, 19 Sep 2011 20:54)

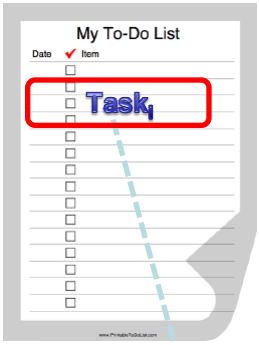
A **Petri Net** is a mathematical modeling language for the description of distributed systems. A Petri net is a directed bipartite graph, in which the nodes represent **transitions** (i.e. events that may occur) and **places** (i.e. conditions). The **directed arcs** describe which places are pre- and/or post-conditions for which transitions. Places may contain a discrete number of **tokens**. Any distribution of tokens over the places will represent a configuration of the net called a marking. A transition of a Petri net may **fire** whenever there are tokens at all input places; when it fires, it consumes these tokens (usually one from each place), and places tokens at the output places of fired transition.



Each task has a responsible person (or unit or facility), who has the task in her to-do checklist

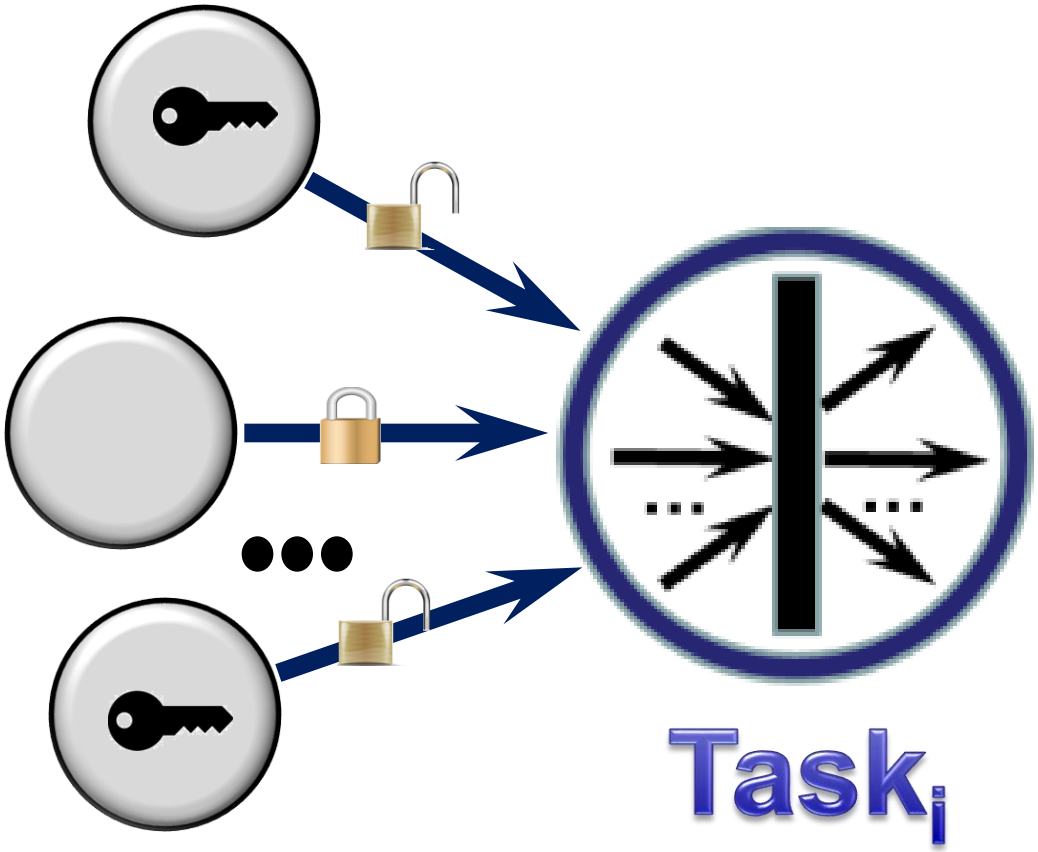


Each task is represented with a transition in the Petri Net model of the military logistics process (hereafter - “business process”



Task$_i$

Preconditions



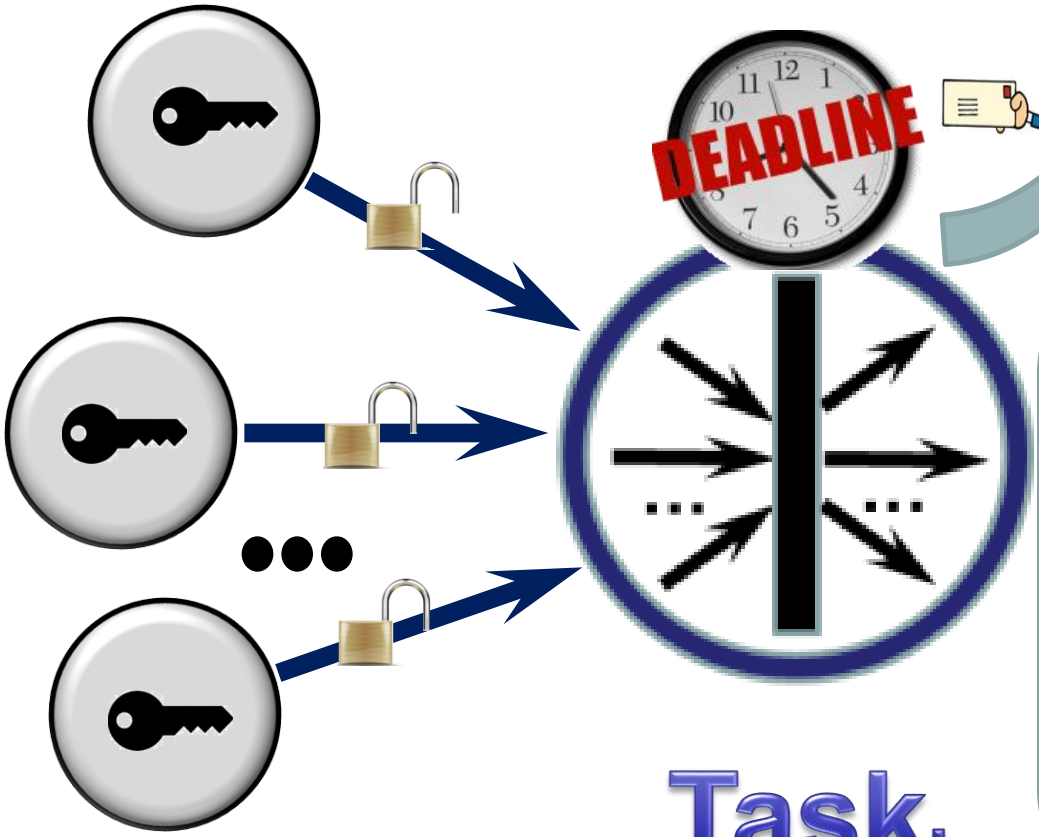
Each task has preconditions (“locks”), which should be unlocked before the task could be performed. A precondition is modeled by a Petri Net place connected with the task-transition. If there is a token (“key”) in some place, then it means that appropriate precondition is unlocked. If all preconditions for some task are unlocked, the task considered to be in an active state (“unlocked”), which means that the responsible person is allowed and expected to perform the actual task

Responsible person

Warning service

Gmail by Google

Preconditions

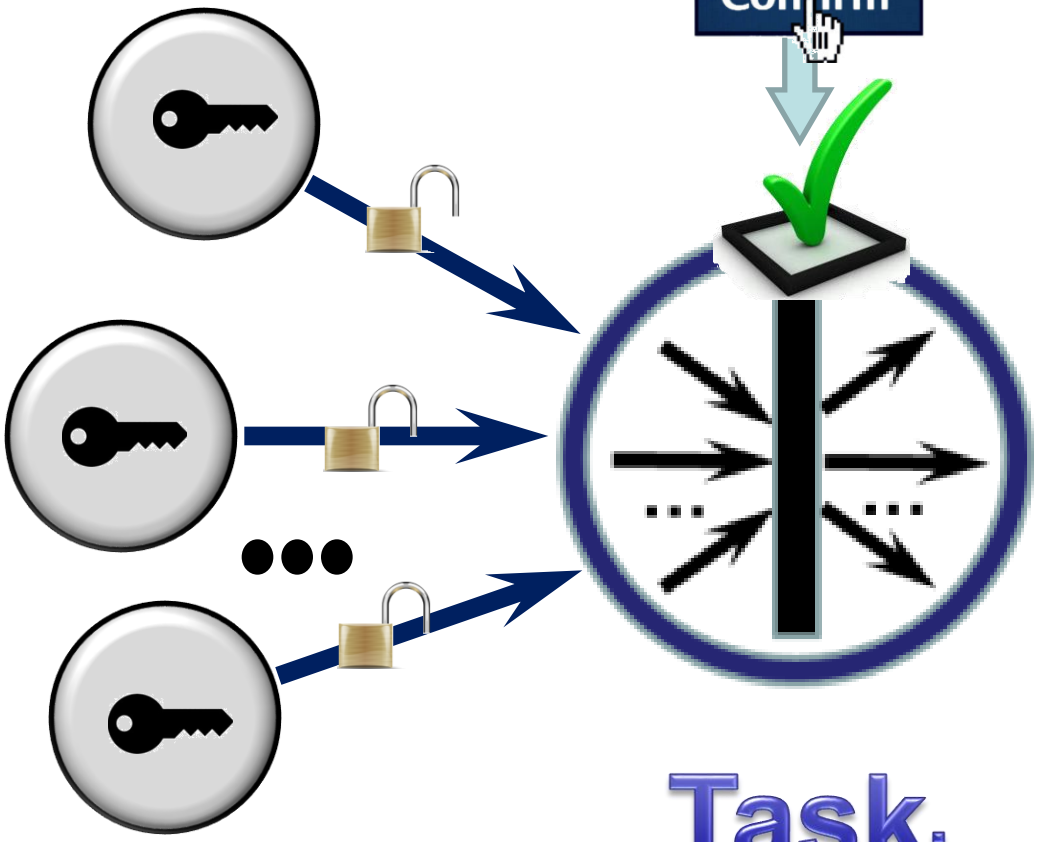


$Task_i$

Each task has certain status at each particular time depending on whether task is locked or unlocked, whether deadline is soon or not, etc. Status is shown to the responsible person via Web interface and in emergent cases a warning (e.g. e-mail, SMS, etc.) is automatically sent to her…

Responsible person

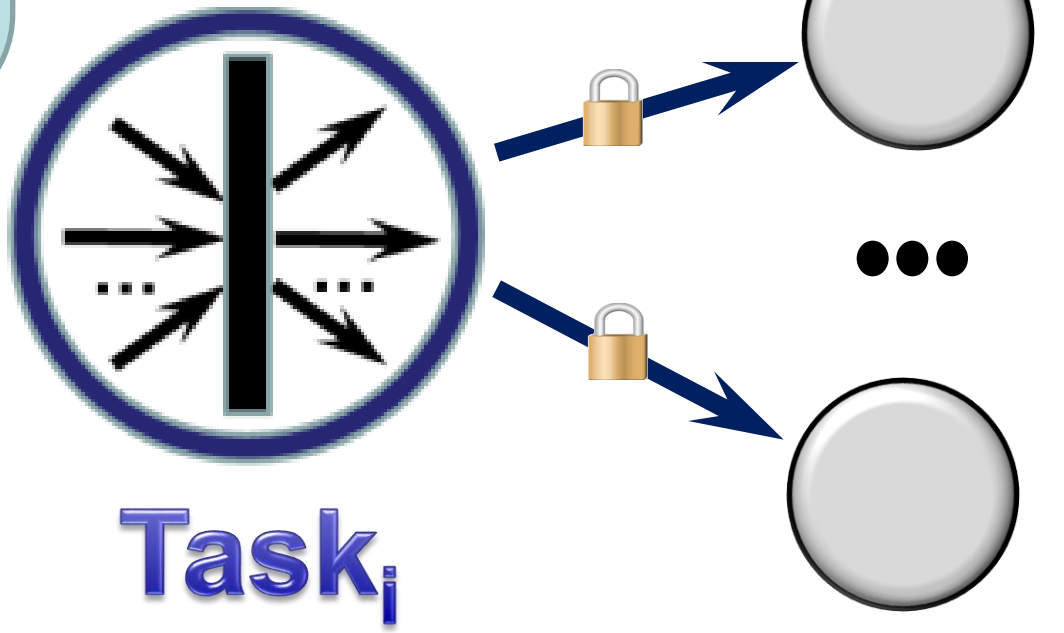
Preconditions

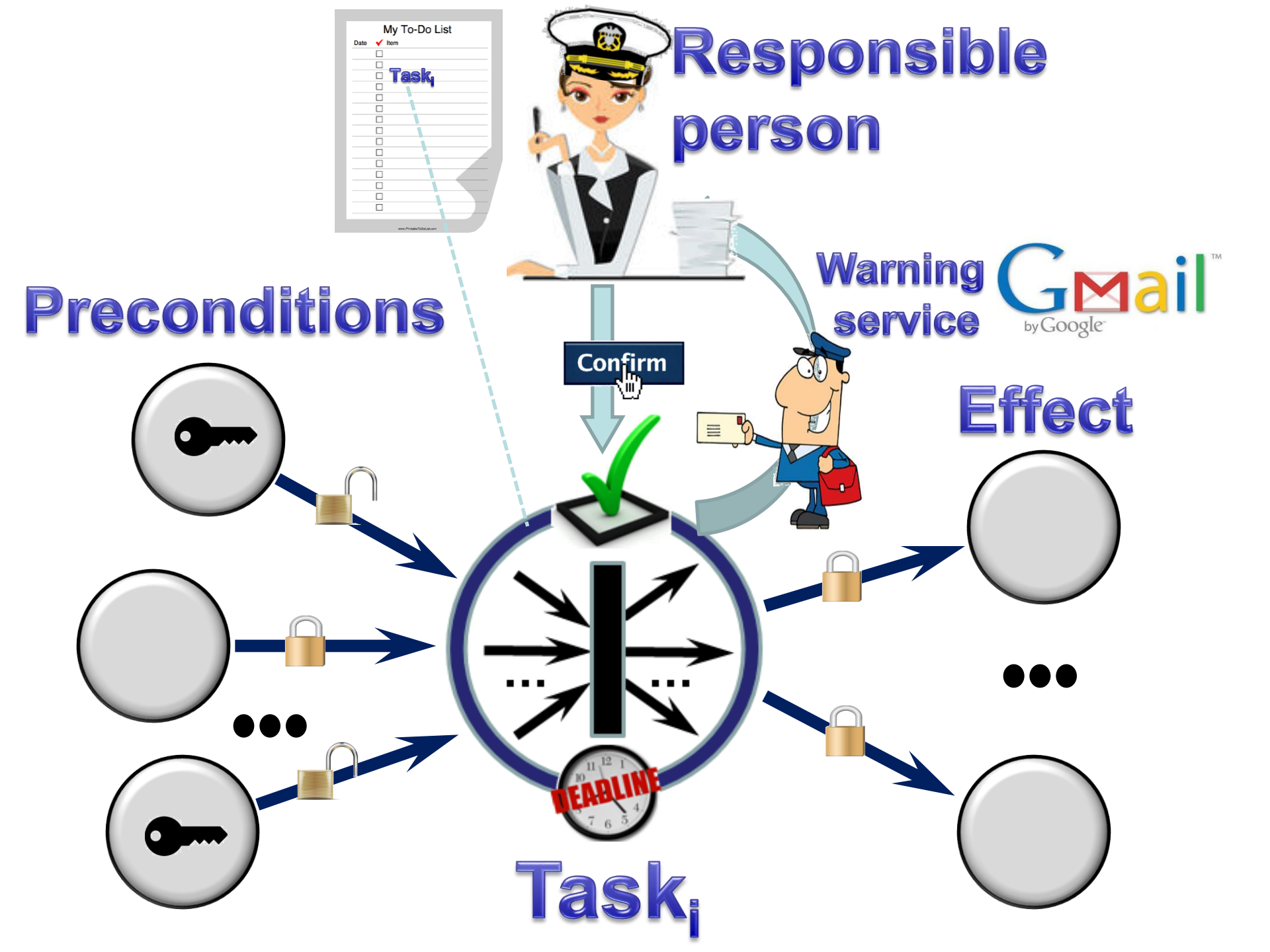


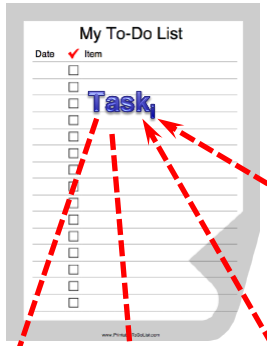
$Task_i$

Unlocked transition is only the necessary condition for it to “fire”. Actual firing process requires also confirmation from the responsible (for the task) person that actual task is done by her. Confirmation via Web interface is sufficient condition for the appropriate Petri Net transition to fire.

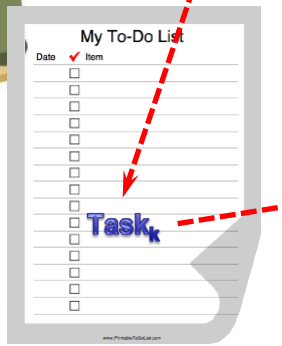
Each task has also an effect (result of the “fire” rule for the Petri Net transition). Effect is modeled by output Petri Net places connected with the task-transition. If the transition fires it removes exactly one token from each input place and sends exactly one token to each output place (unlocking by this other depending tasks).

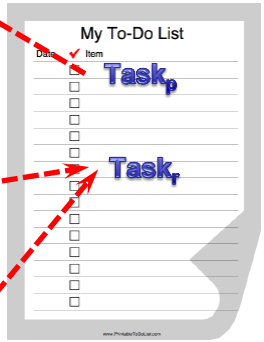


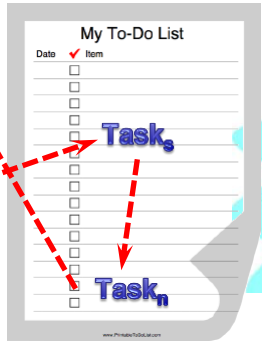

My To-Do List
Date
Item
Task$_i$
Responsible person
Warning service
Gmail by Google
Preconditions
Confirm
Effect
DEADLINE
Task$_i$

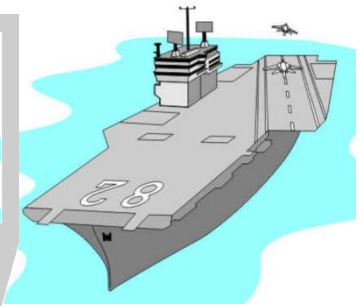


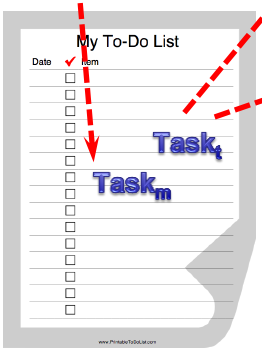
Semantics of such link:
"depends_on"
or "locked_by"



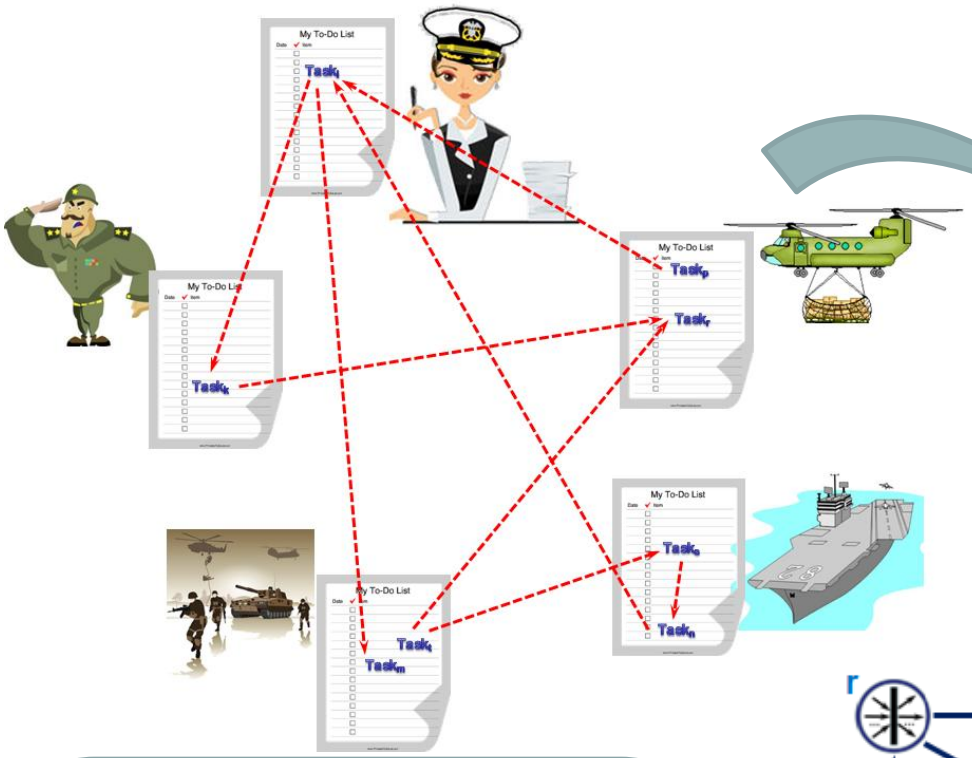






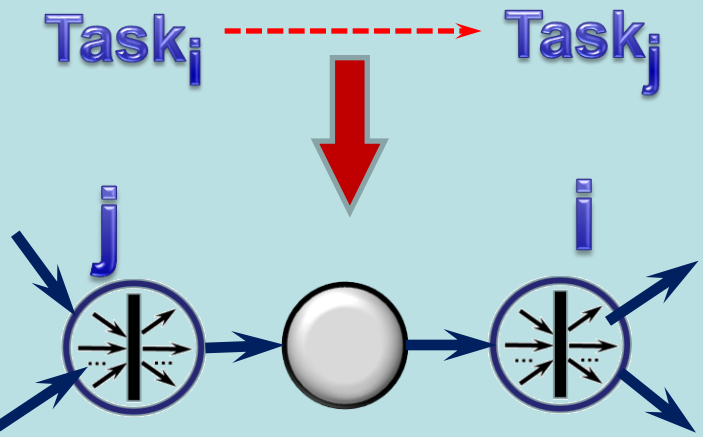


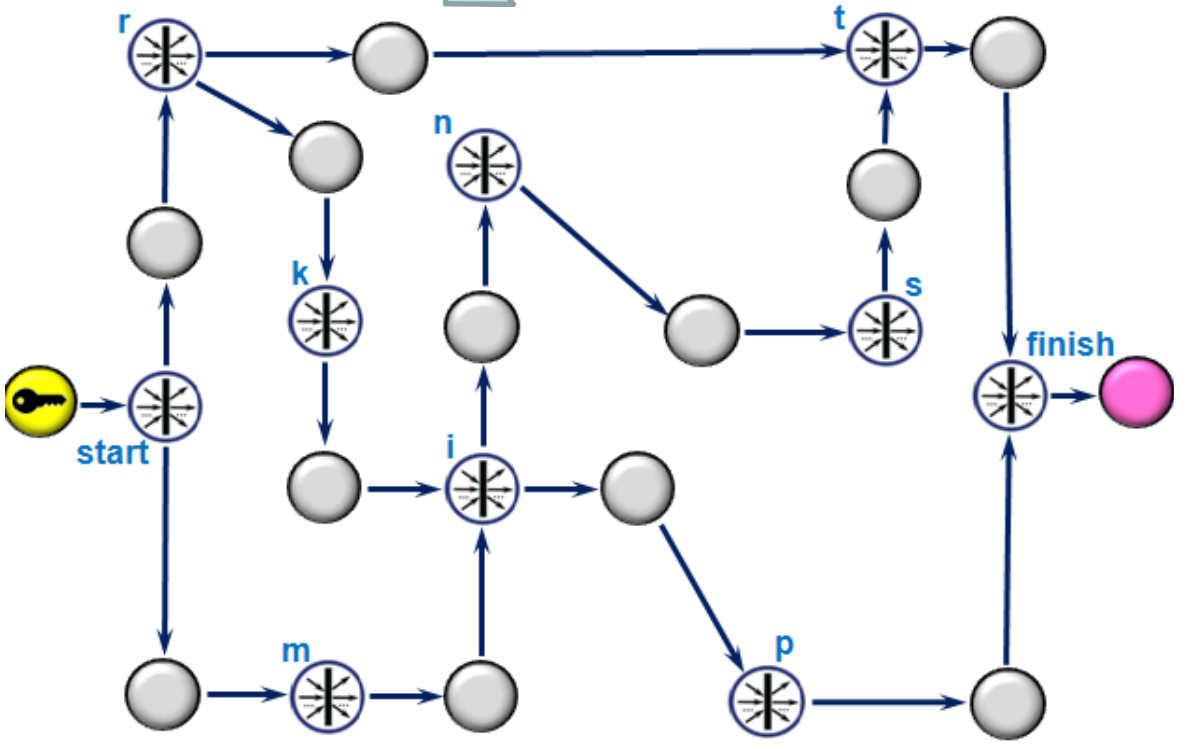
Each business process is a graph of individual tasks connected with dependency relations

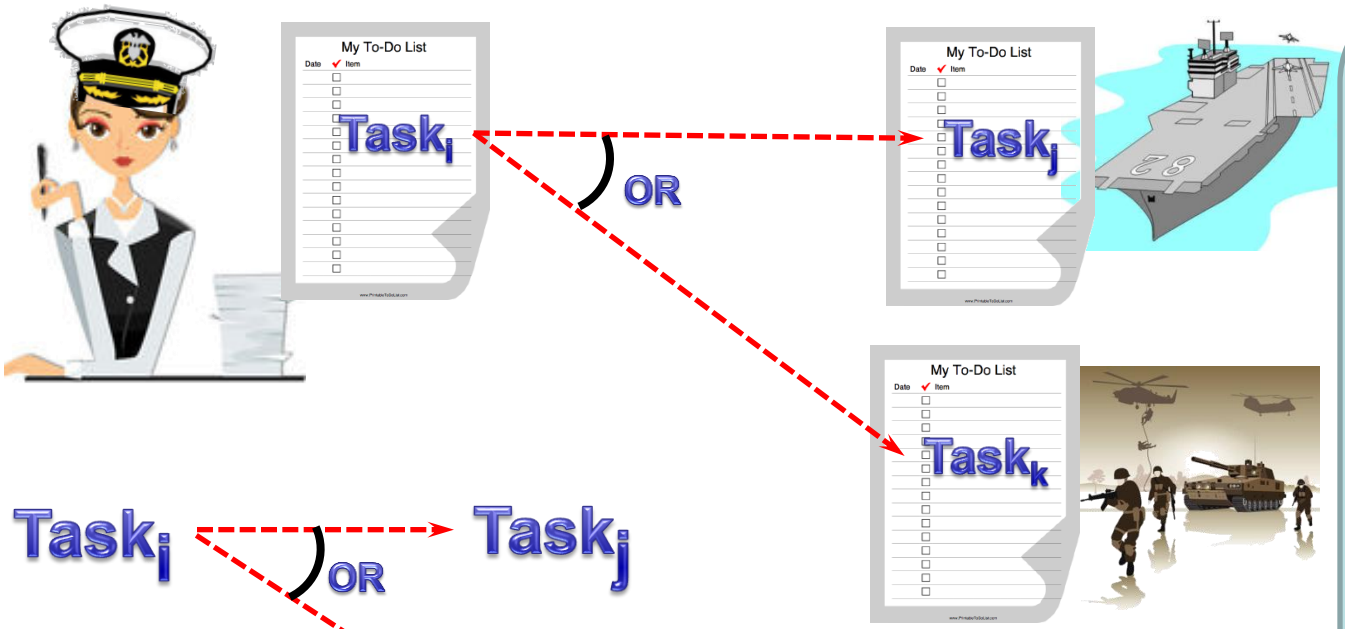


There is a simple transformation from a business process (**AND-OR**) graph to the appropriate Petri Net model of it. Therefore the process manager can easily create and edit business process in the form of graph

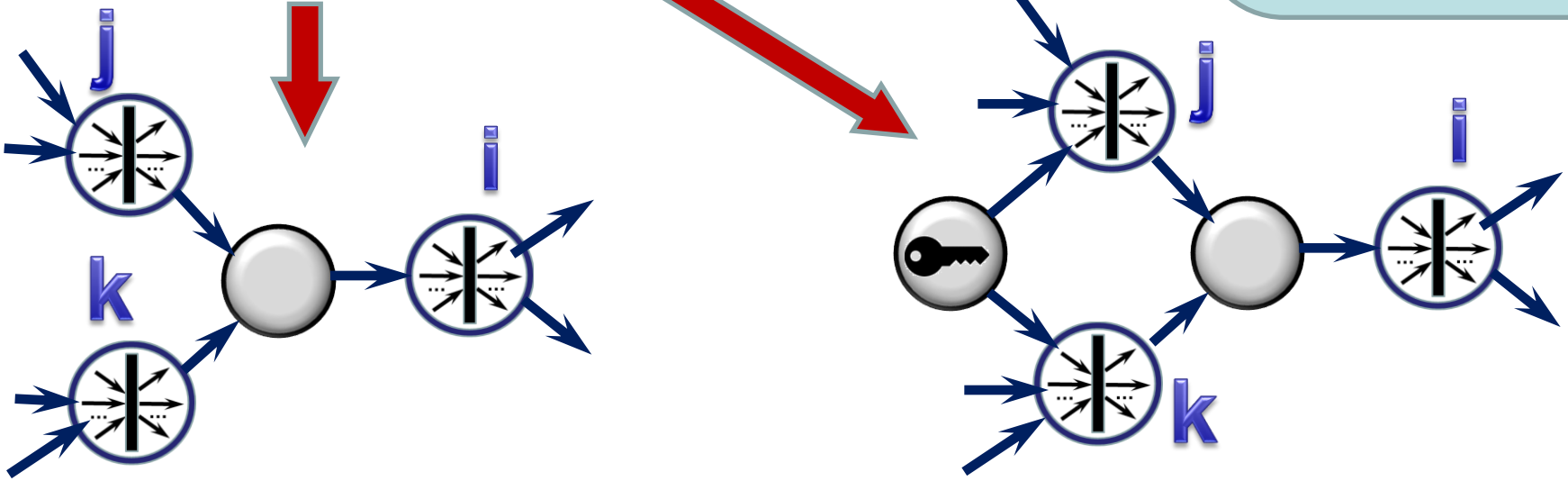
Transformation rule







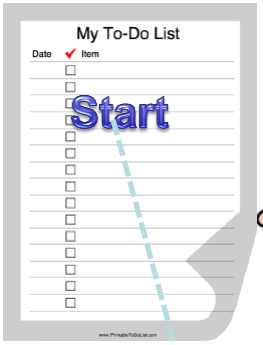
OR relation between tasks means that for some task to be unlocked is sufficient that at least one of two or more other tasks are done. Sometimes use of **Exclusive-OR** is better for the economy of process resources



Transformation rule for the OR relation between tasks

Transformation rule for the exclusive OR relation

Task "Start" (just one)

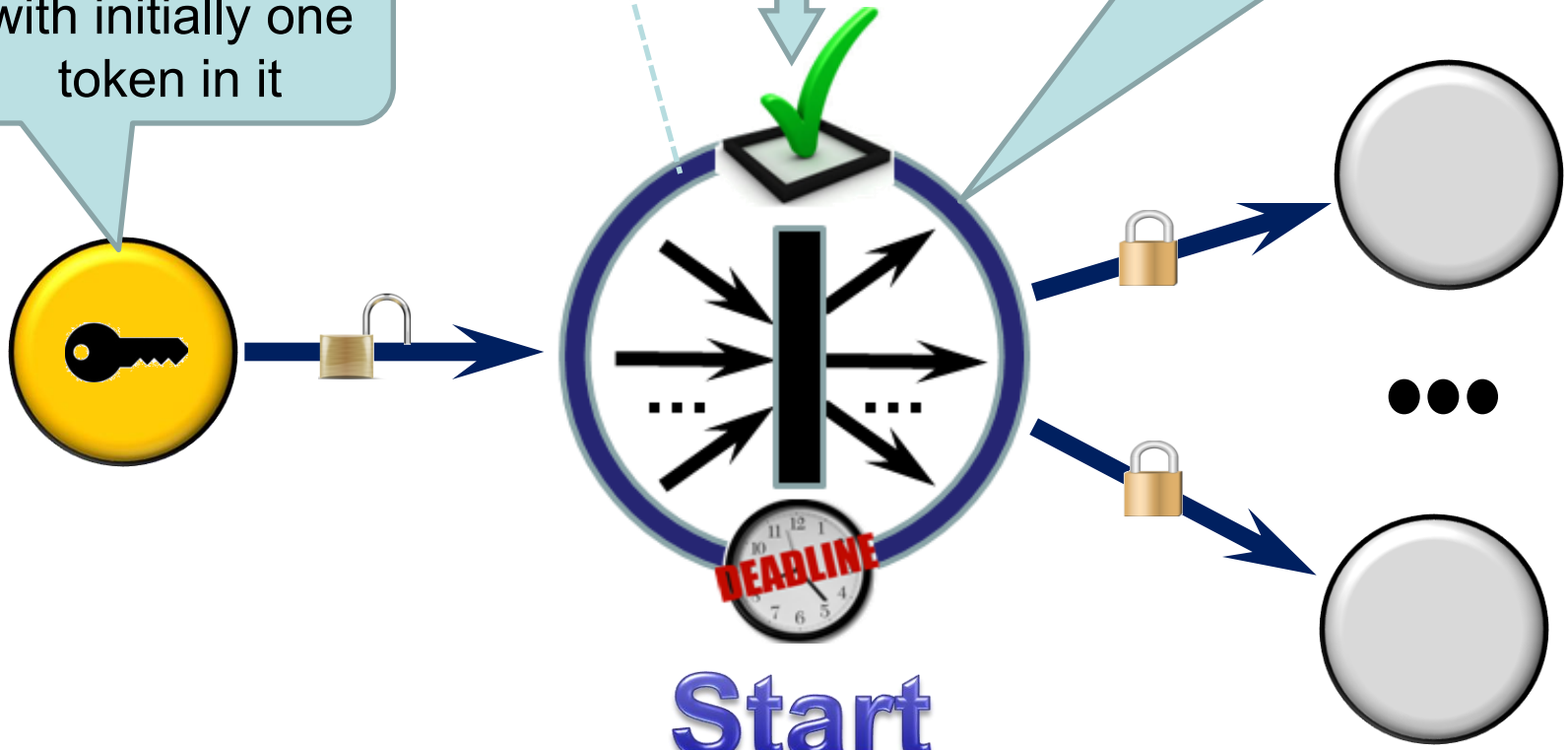


Process Manager

Unlocks all the tasks, which are not dependent on any other tasks. Confirmation of the "Start" task by process manager means official start of the business process.

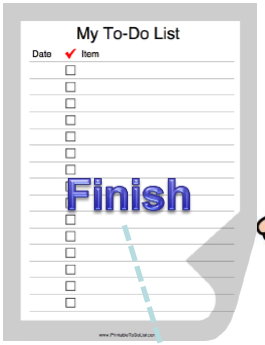
One input place with initially one token in it

Confirm



Start

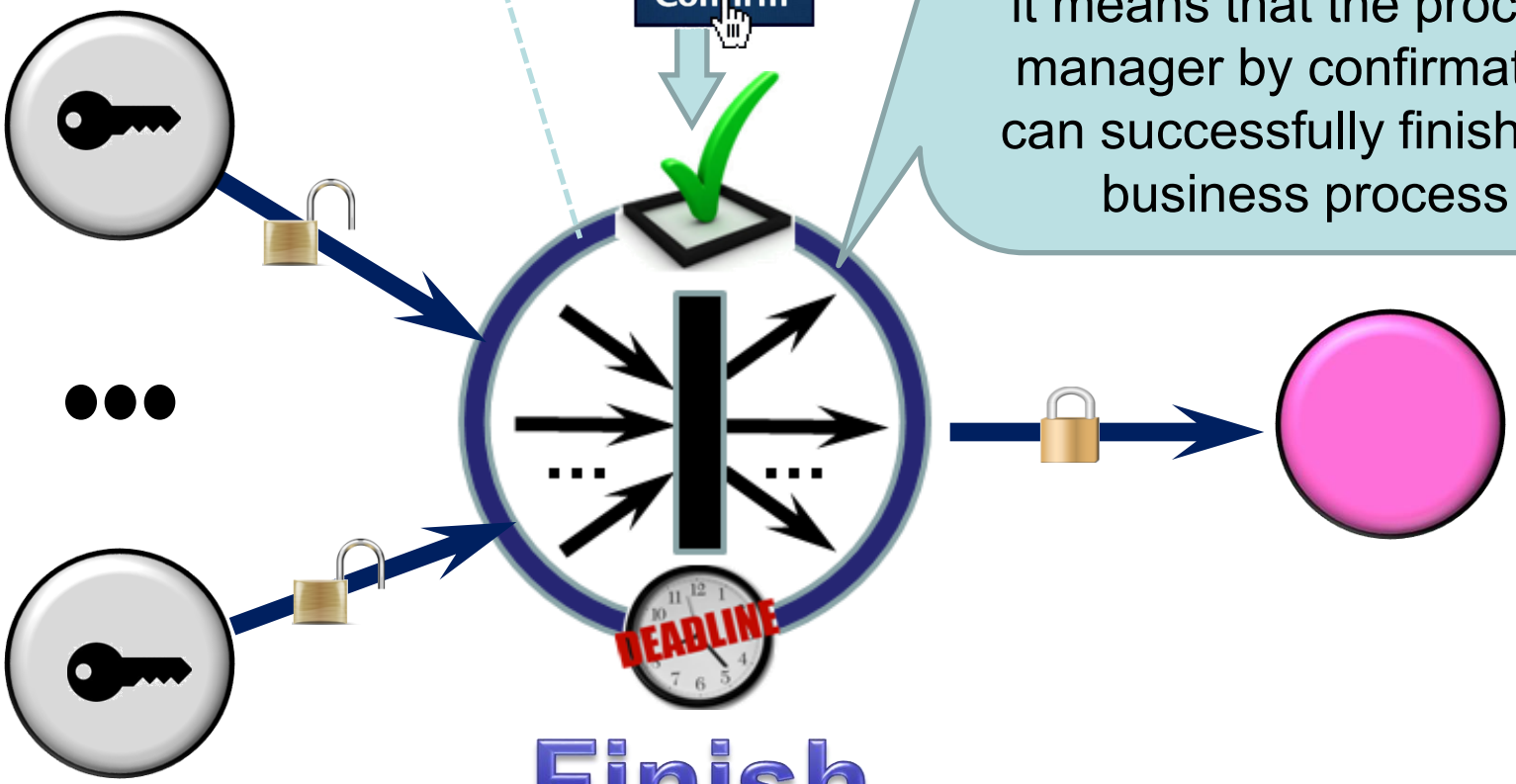
Task "Finish" (just one)



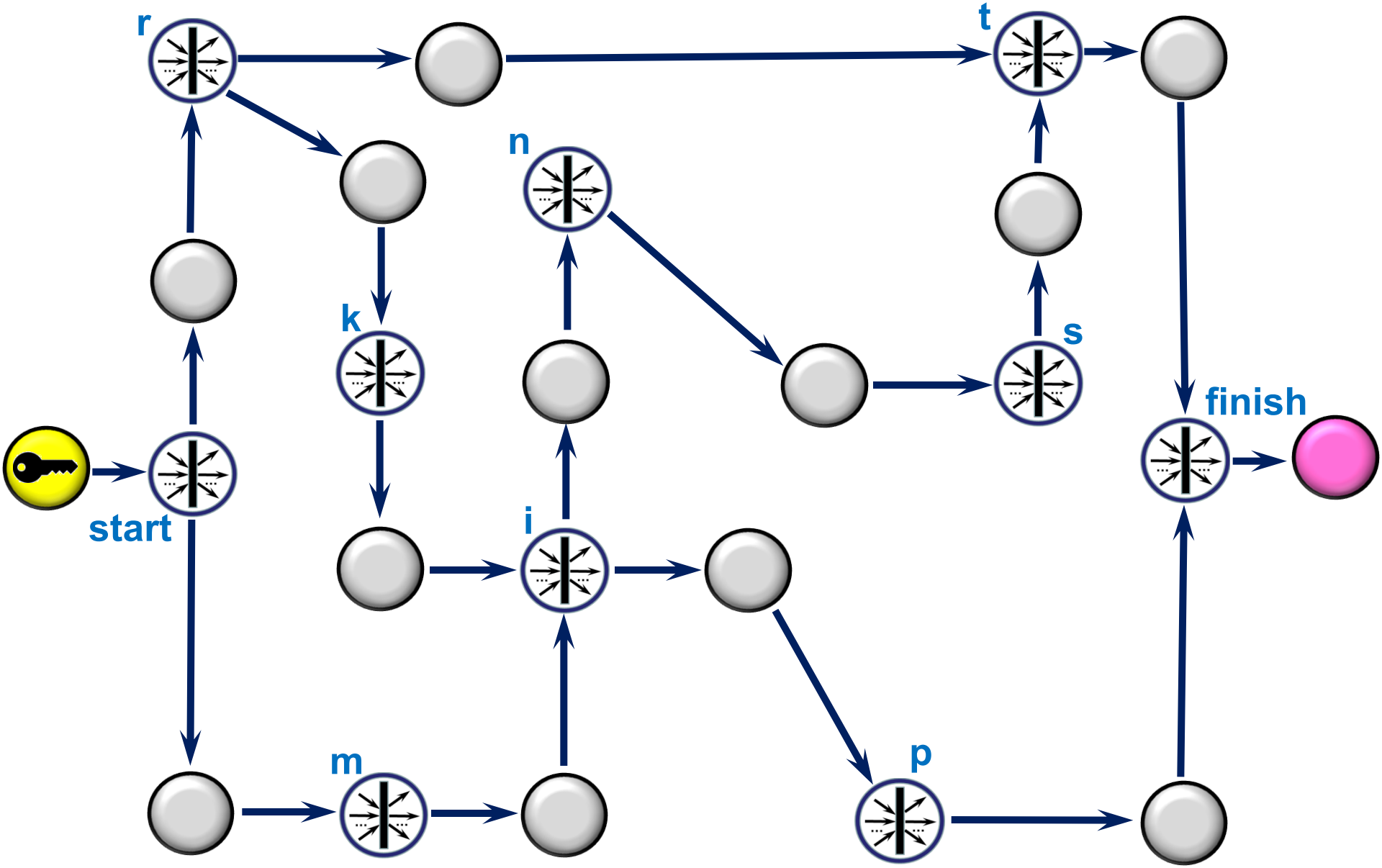
Process Manager

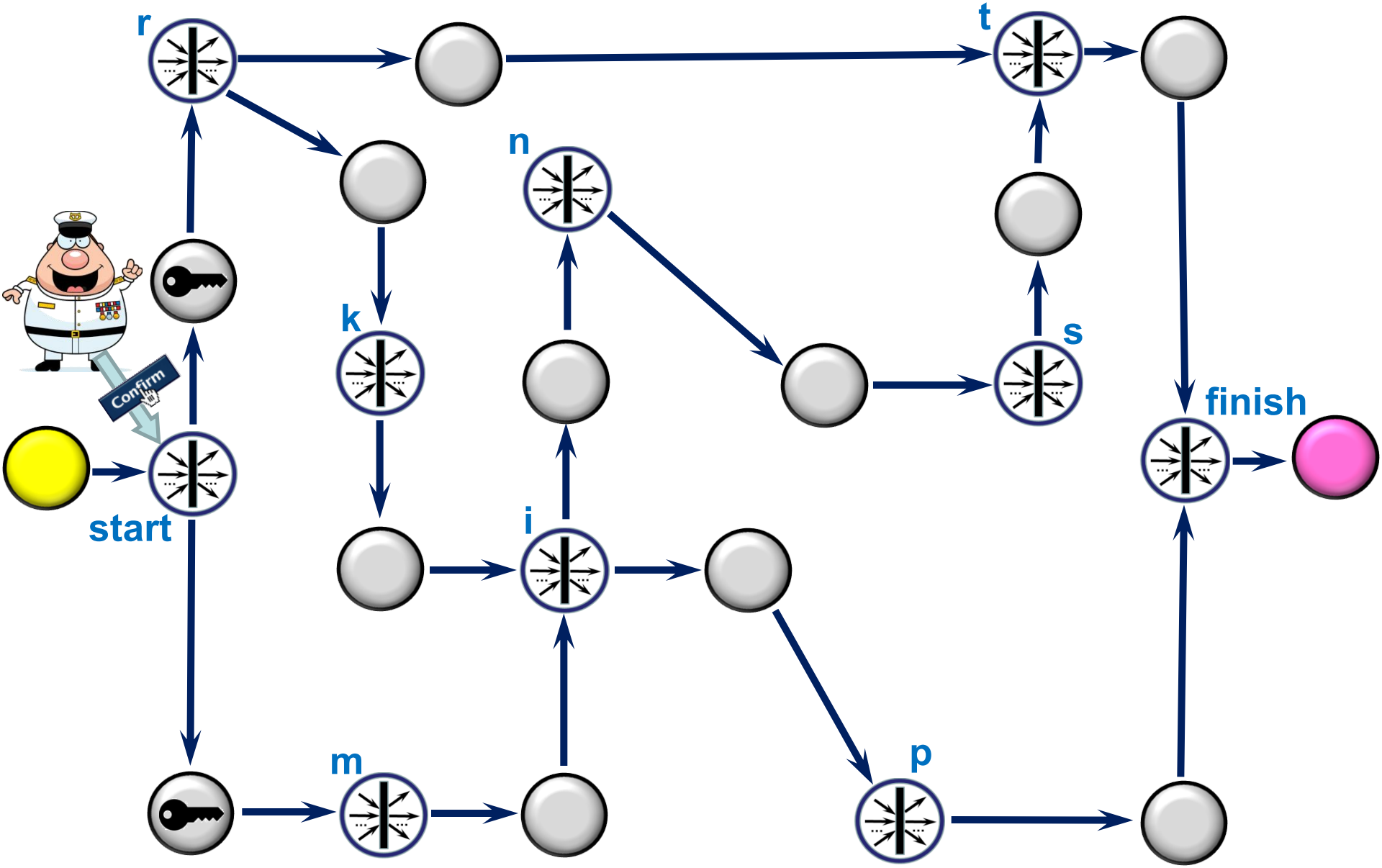
Confirm

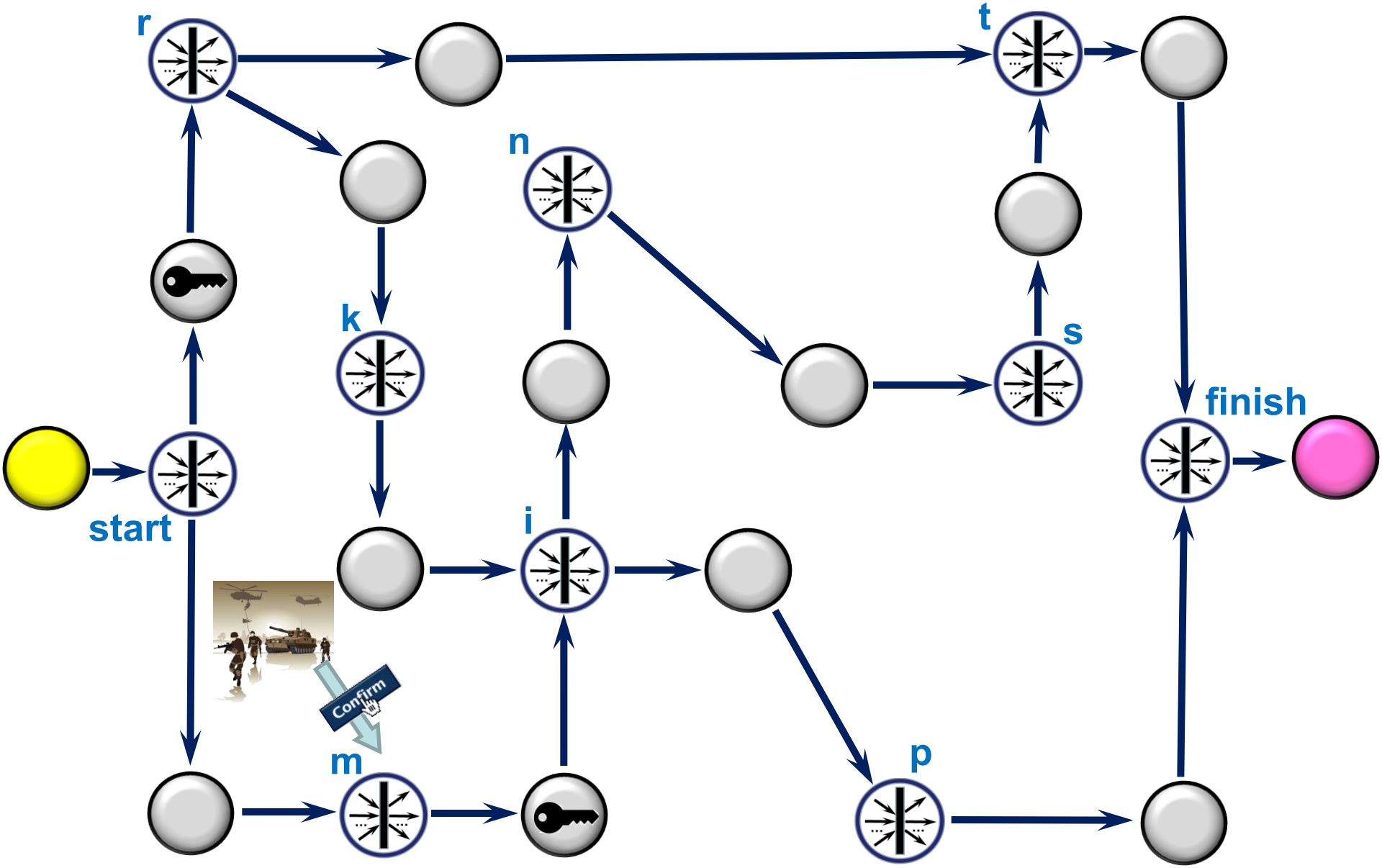
All the tasks, on which none of other tasks are dependent, are becoming preconditions for the "Finish" task. If this task in unlocked it means that the process manager by confirmation can successfully finish the business process

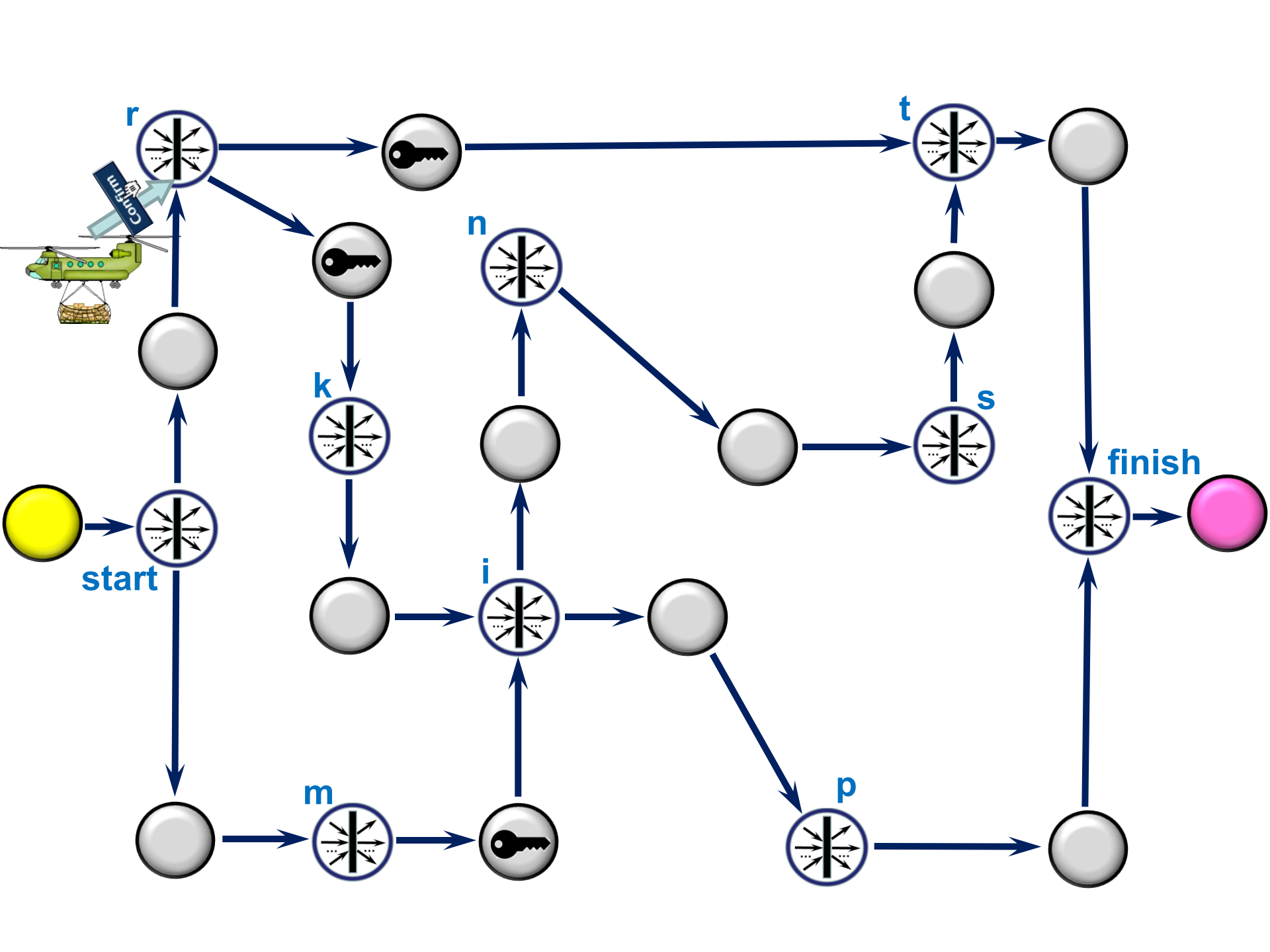


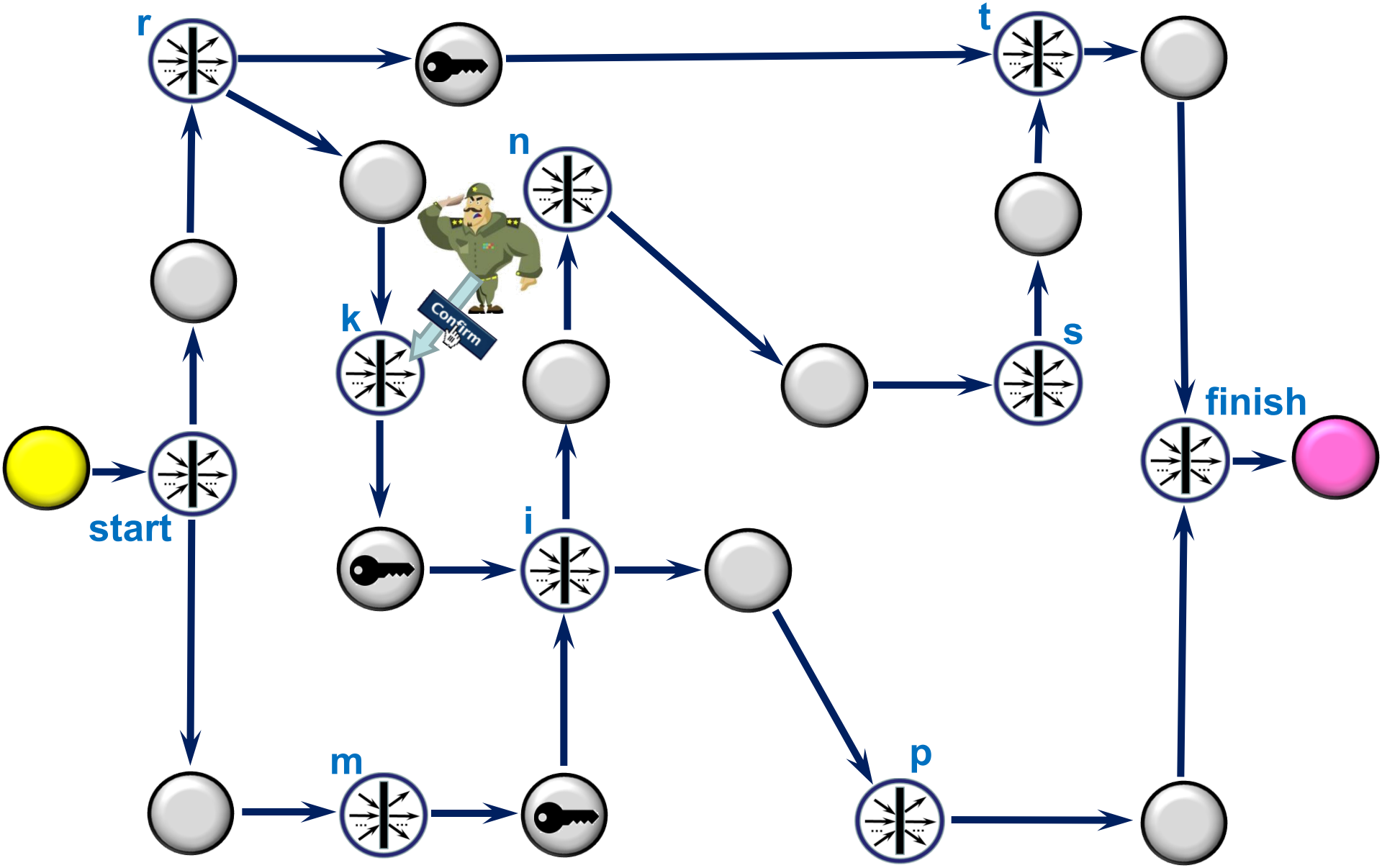
Finish

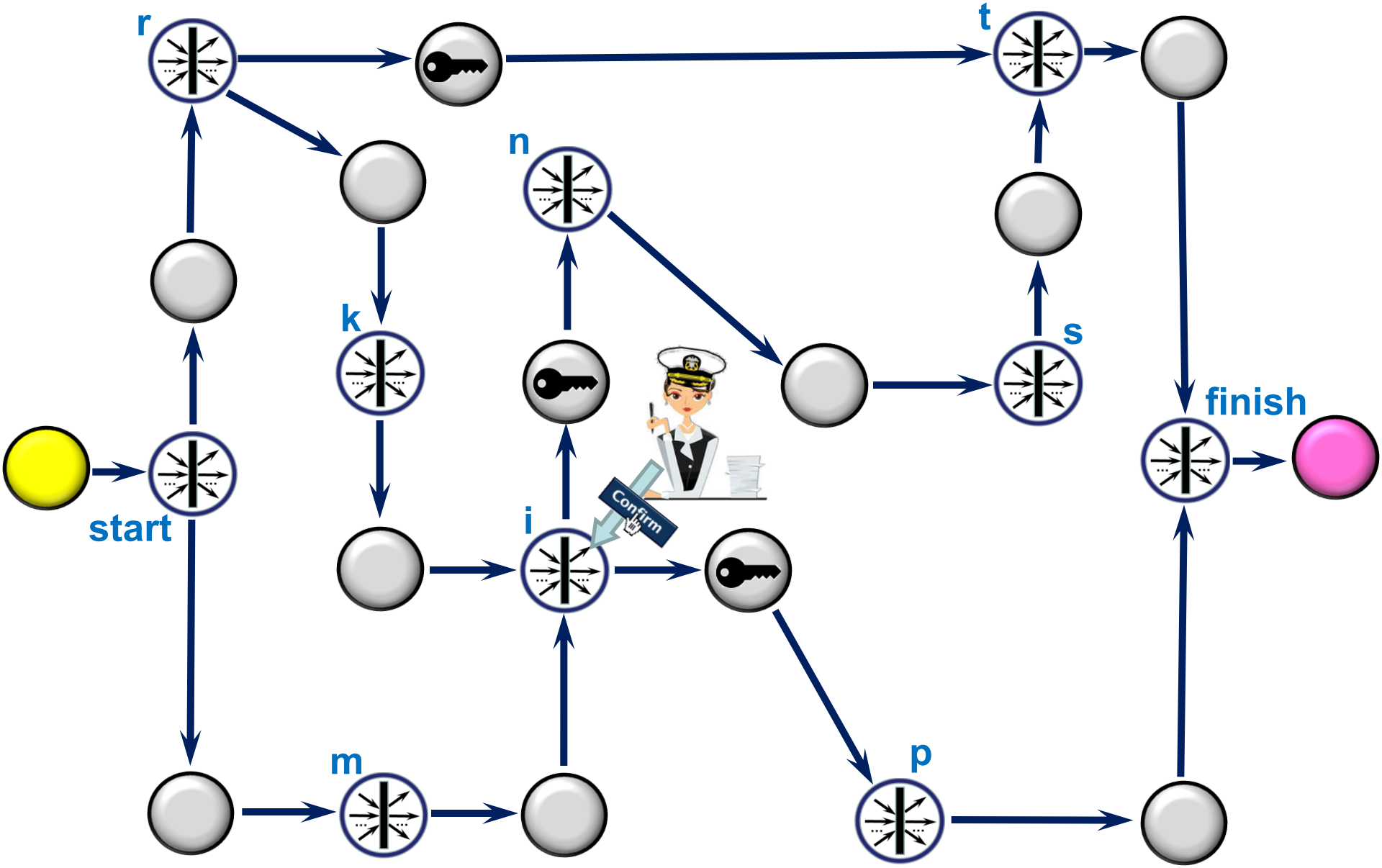

r
t
n
k
s
finish
start
i
m
p

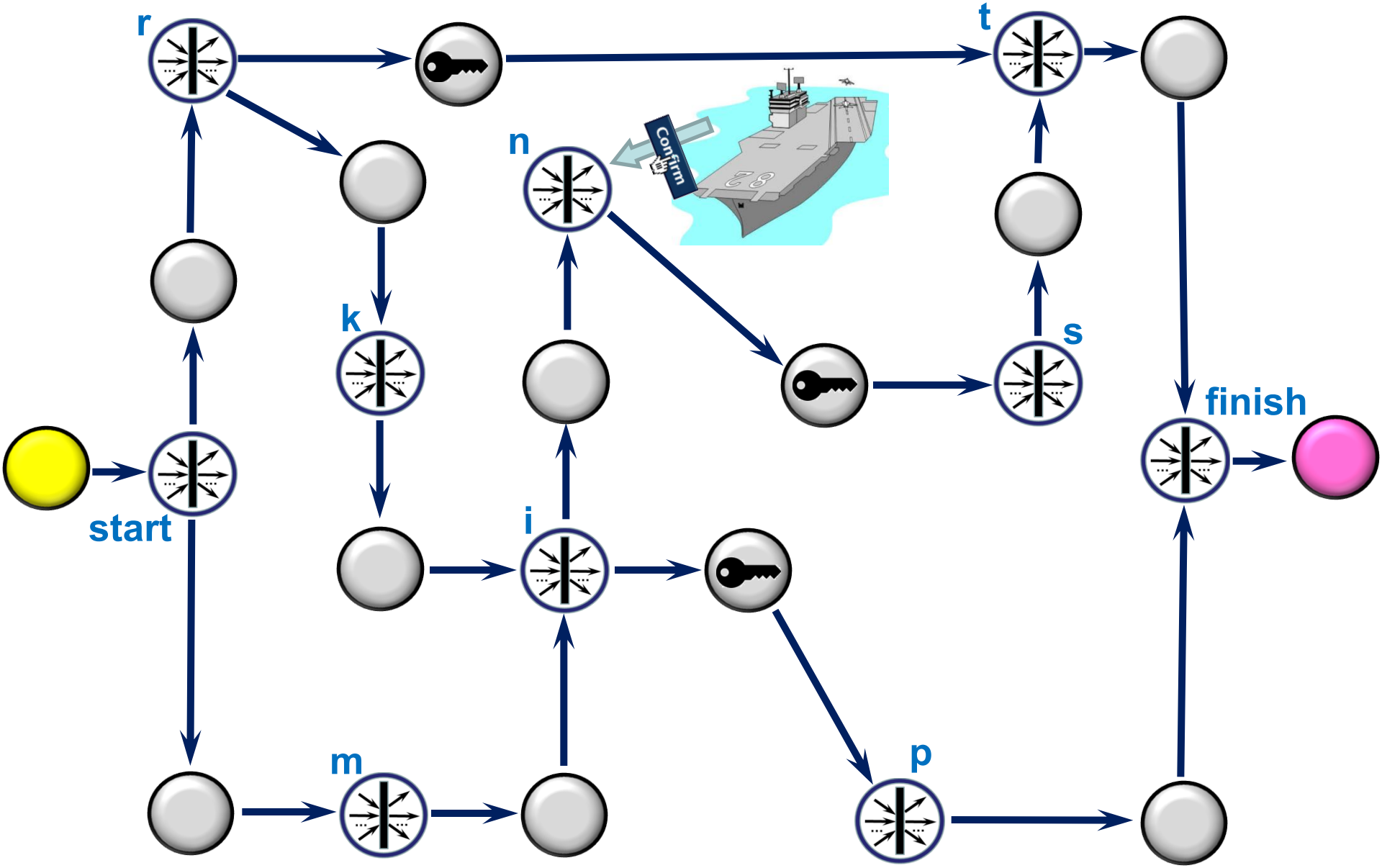

r
t
n
k
s
finish
start
i
m
p
Confirm

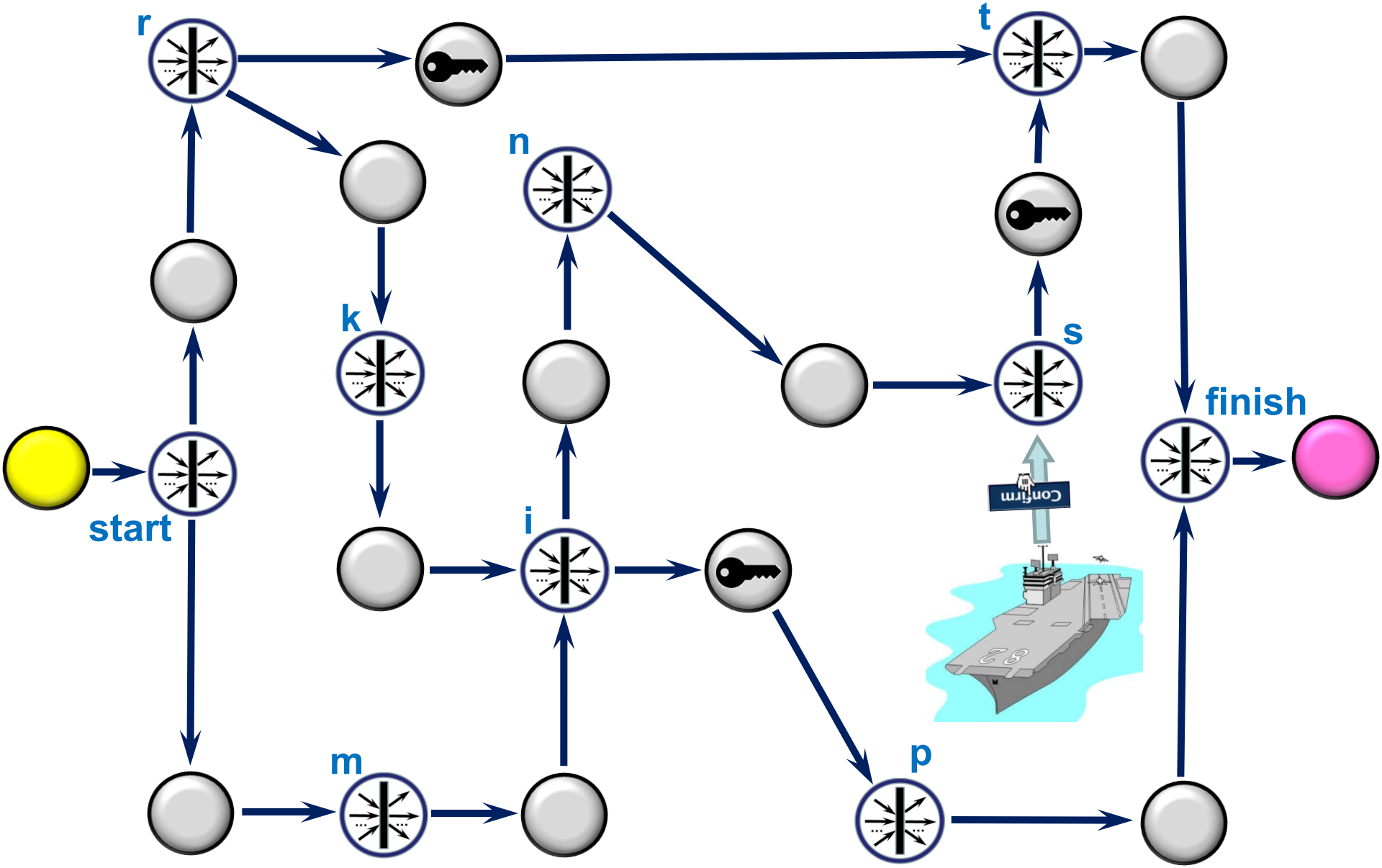

r
t
n
k
s
finish
start
i
Confirm
m
p

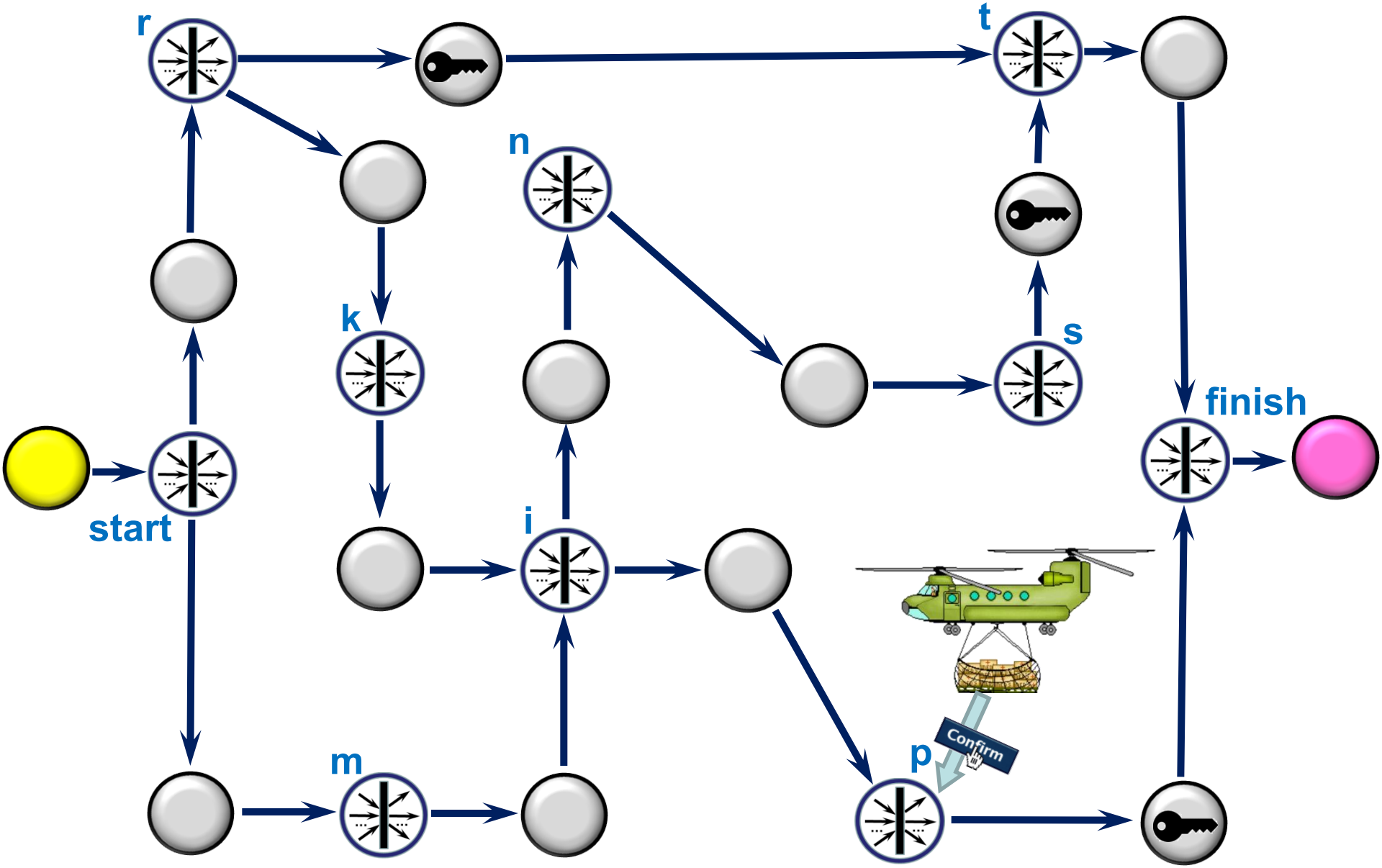

r
t
n
k
s
finish
start
i
m
p
Confirm

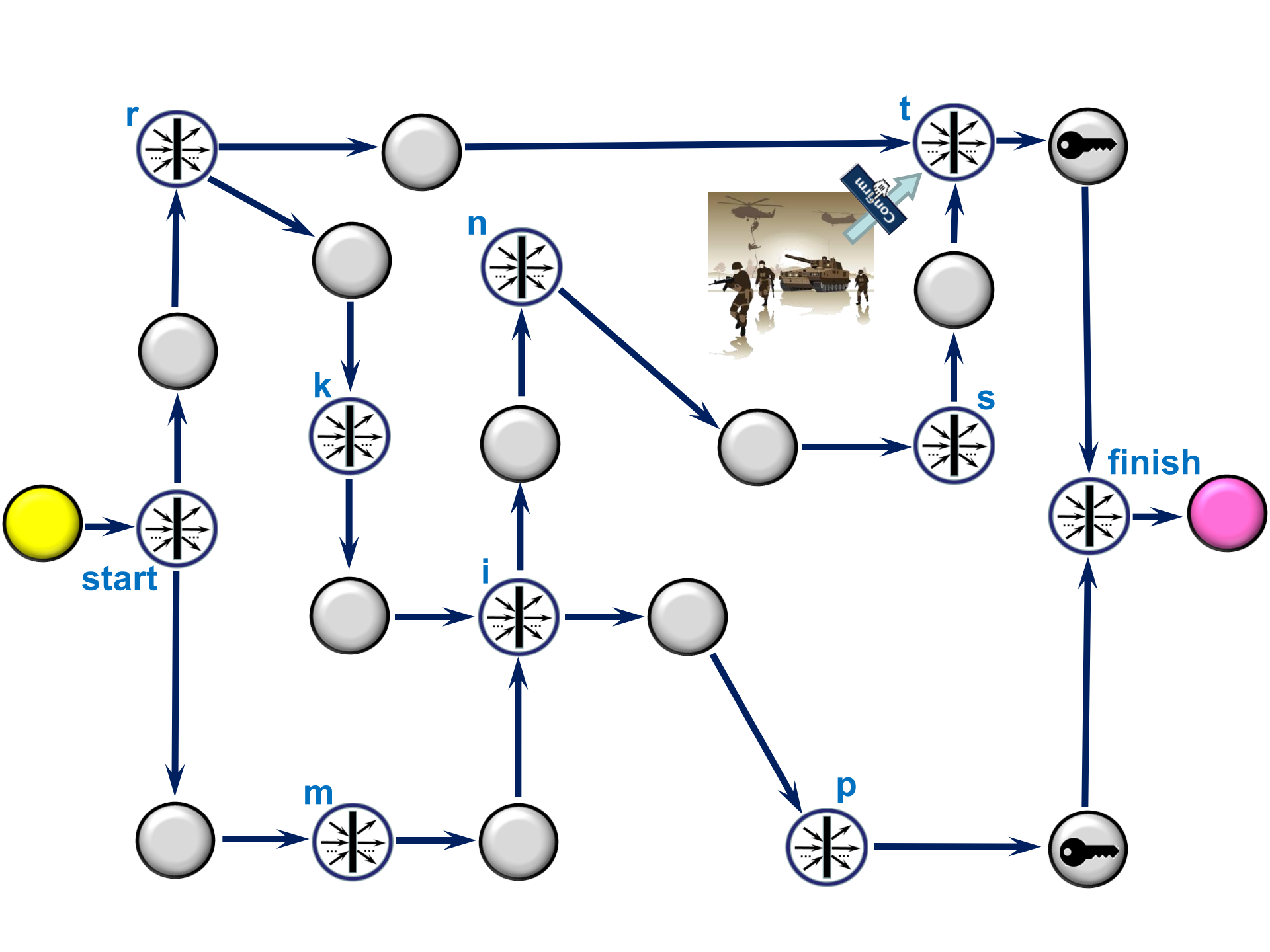

r
t
n
k
s
finish
start
i
m
p
Confirm

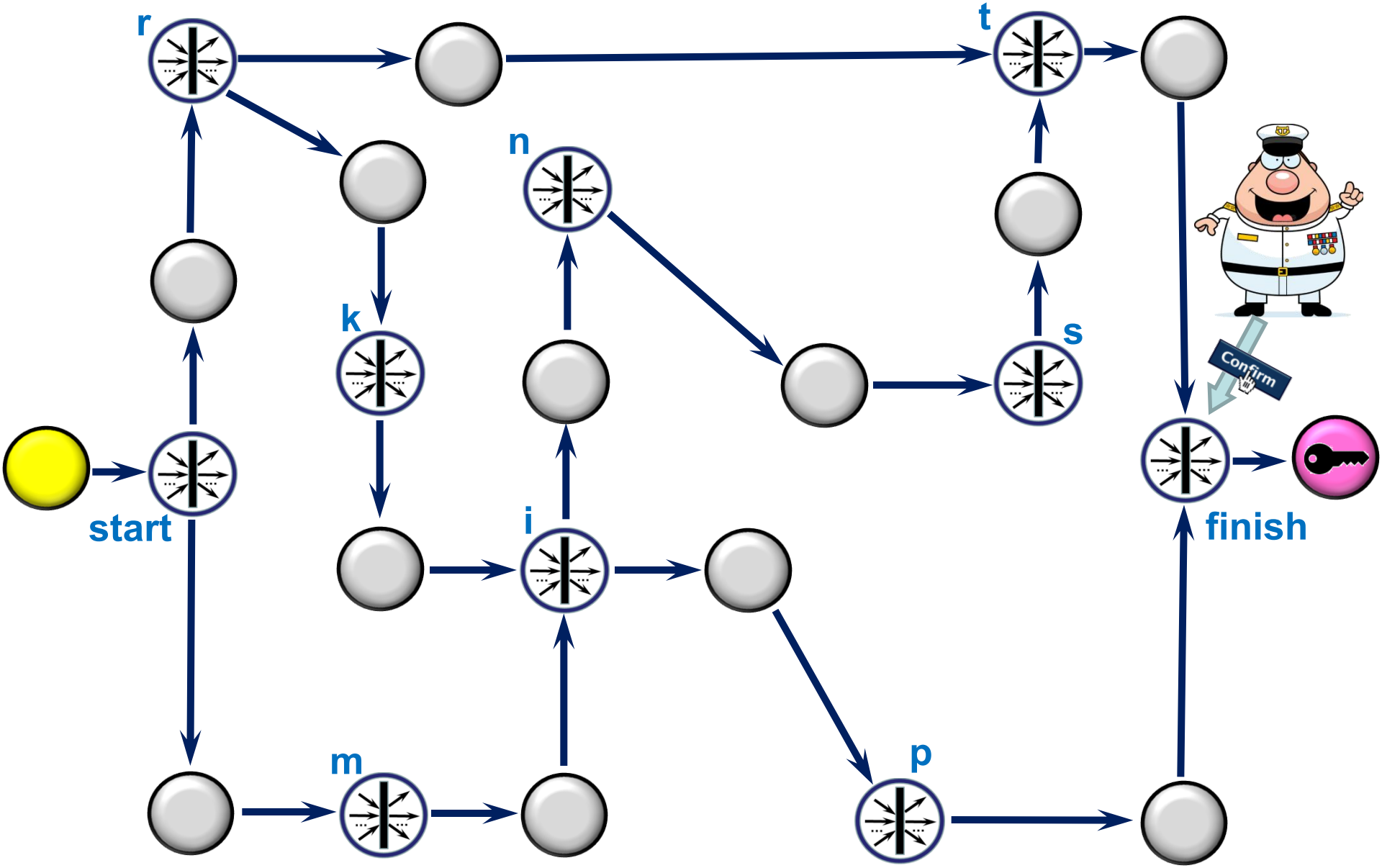

r
t
n
k
s
start
i
finish
m
p
Confirm

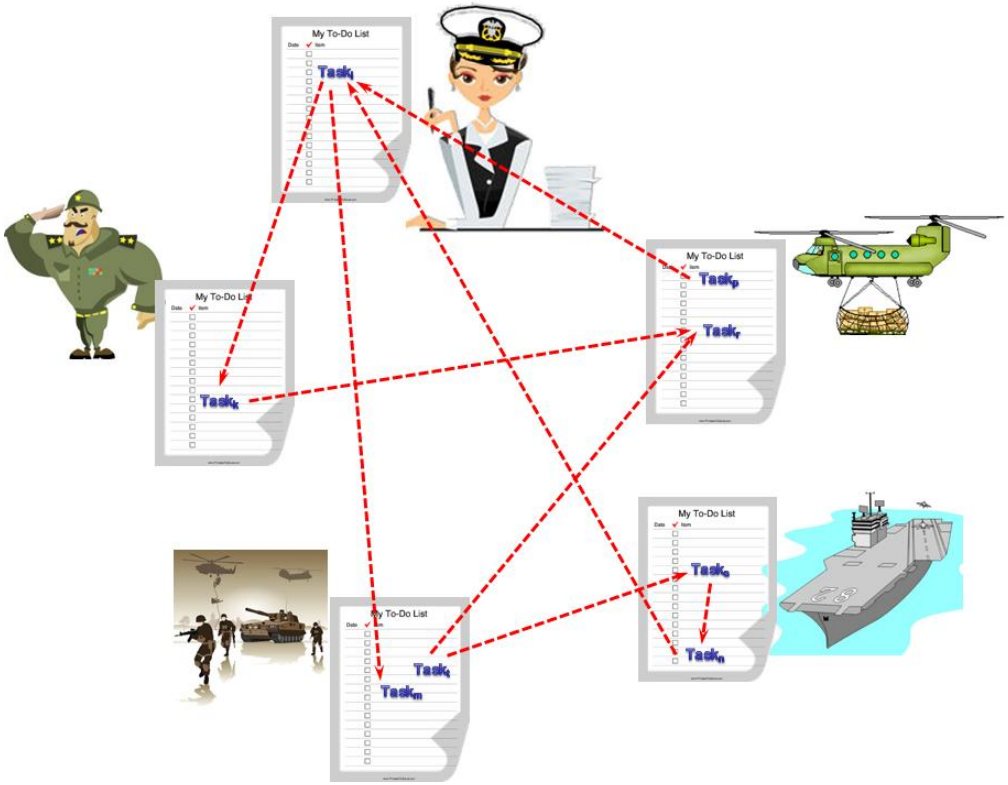

r
t
n
Confirm
k
s
finish
start
i
m
p


r
t
n
k
s
finish
start
i
m
p
Confirm


r
t
n
k
s
finish
start
i
m
p
Confirm


r
t
n
k
s
start
i
finish
m
p
Confirm

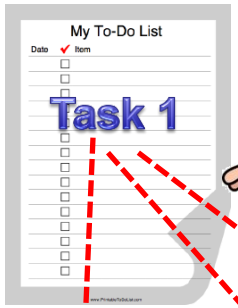

r
t
n
k
s
start
i
m
p
finish
Confirm

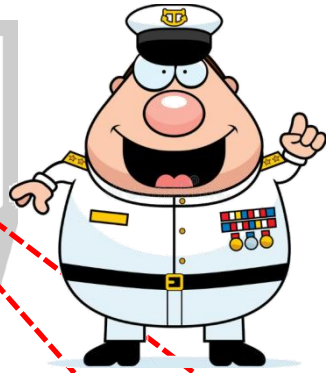


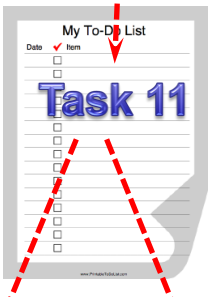
Based on a business process graph it will be possible to calculate also the importance (rank) of each task, utilizing for example Google Page Rank algorithm. The analogy here is as follows: “A task will be more important if important tasks are locked by it.”

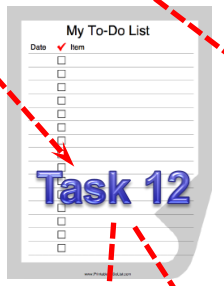
Task rank helps to order tasks in to-do checklists according to importance. It can be used also in warning decisions, for example, if some task has a very hot deadline and it also has high rank in the same time (i.e. “many other tasks are pending because of this task”), then the automated “push”-warning of the responsible person is applied

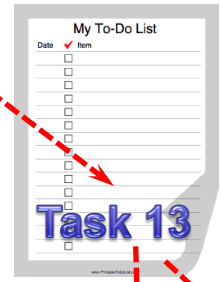
Nested processes
(task as a business process)

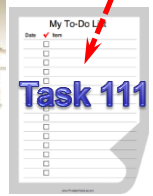


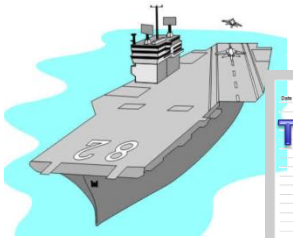


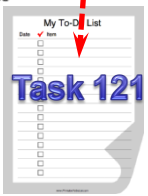


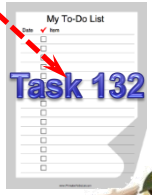




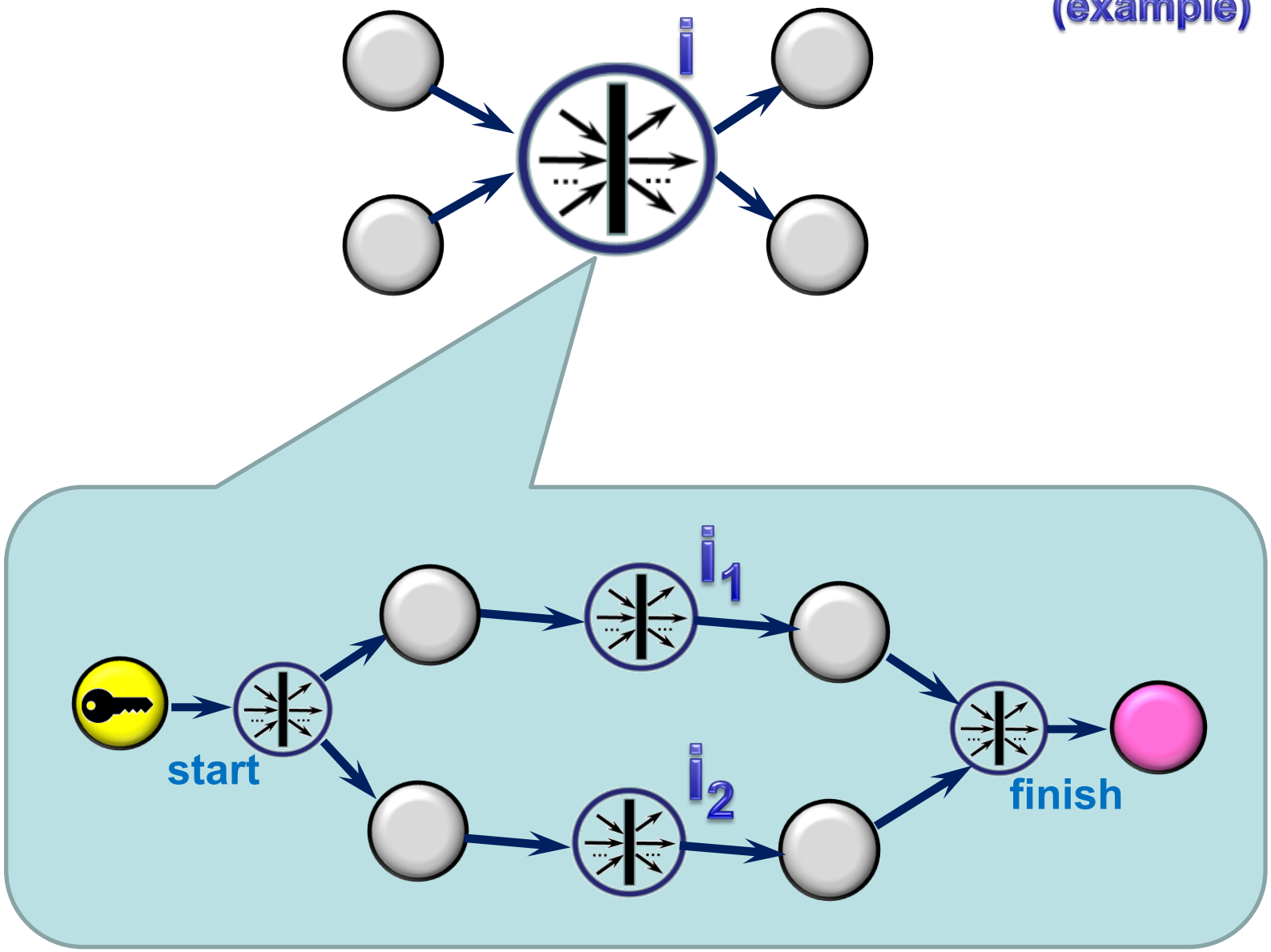




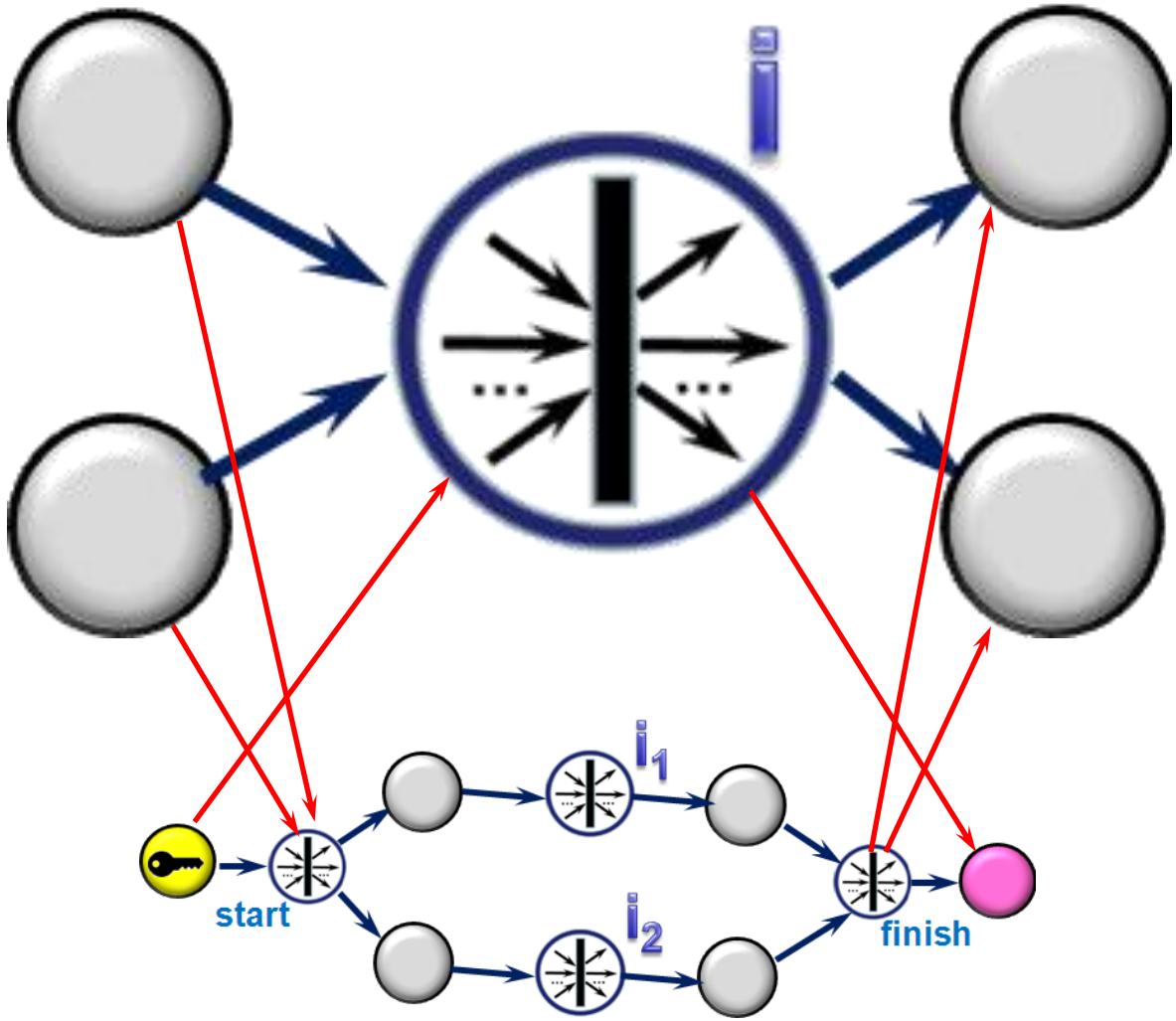




Nested process
(example)



Nested process
(example)



First Demo
(Admiral 1.0)

We put the video, with the demonstration to the Web folder:
http://www.cs.jyu.fi/ai/admiral/videos/

First look at main-final.avi and then email-final.avi.

The Web link to the application is here: http://ub1.ad.jyu.fi/ .
It shows some sample scenario modeled with Admiral 1.0 and which you can try to run yourself via one or several Web interfaces

Note that User mikko has login "mikko" and password "mikko". All the experts have password "a" (as mentioned in the video).

Now it is time for
(Admiral 2.0)

Some slides from our recent Kick-Off (May, 2018)

An Immune System for Autonomous Supply-Chain Management and Logistics Against Data Poisoning

Kick-off meeting of the NATO SPS G5511 Project "Cyber Defense for Intelligent Systems"

Vagan Terziyan

Faculty of Information Technology, University of Jyväskylä

URL:http://www.mit.jyu.fi/ai/vagan

vagan@jyu.fi

NATO OTAN

This workshop is supported by: The NATO Science for Peace and Security Programme

Jyväskylä, May 24, 2018

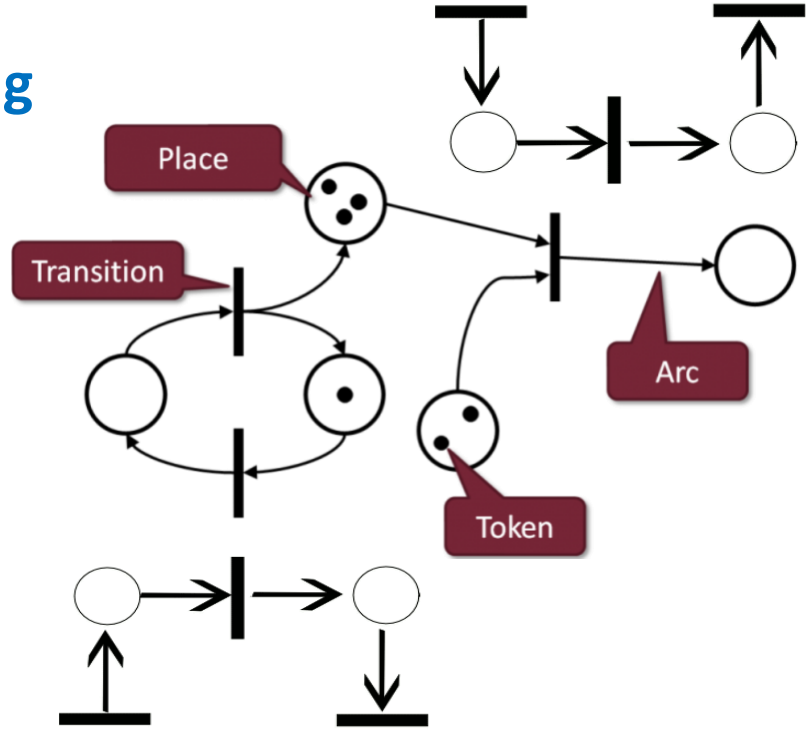
Why Petri Nets ?

*"**Supply networks are discrete event dynamic systems** consisting of suppliers, manufacturers, distributors, and customers. It is a difficult and challenging task to model such a complex system. Recently, characterized as being capable of model parallelism and synchronization, Petri Nets have attracted great attention for modelling and studying a supply network."*

X. Zhang, Q. Lu & T. Wu (2010). **Petri-Net based Applications for Supply Chain Management: An Overview**. *International J. of Production Research*, 49:13, 3939-3961. doi: 10.1080/00207543.2010.492800

Petri Nets are used exactly for modelling of discrete event dynamic systems !

The formalism of Petri Nets excels at modeling concurrency and synchronization, resource constraints and loosely dependent subsystems. There are efficient analysis (model checking) tools for Petri Nets, capable of e.g. finding deadlocks, deciding the reachability of states with given properties, proving the conservation of resources or the cyclic nature of a sequence of events.



However, we need smarter Petri Nets,

which include also a decision-making component

See some samples of related work:

Expert Systems with Applications
Volume 42, Issue 23, 15 December 2015,
Pages 9307-9317
ELSEVIER

Context-adaptive Petri nets: Supporting adaptation for the execution context

Estefanía Serral , Johannes De Smedt , Monique Snoeck , Jan Vanthienen

https://doi.org/10.1016/j.eswa.2015.08.004

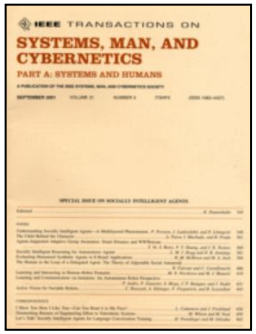
A self-adaptive software system (e.g., a Petri Net model) is one that can autonomously modify its behavior at runtime in response to changes in the system and its environment. This is quite important for military logistics and supply-chain management.

IEEE Transactions on Systems, Man, and Cybernetics: Systems (Volume: 46, Issue: 4, April 2016)

Modeling Self-Adaptive Software Systems With Learning Petri Nets

DOI: 10.1109/TSMC.2015.2433892

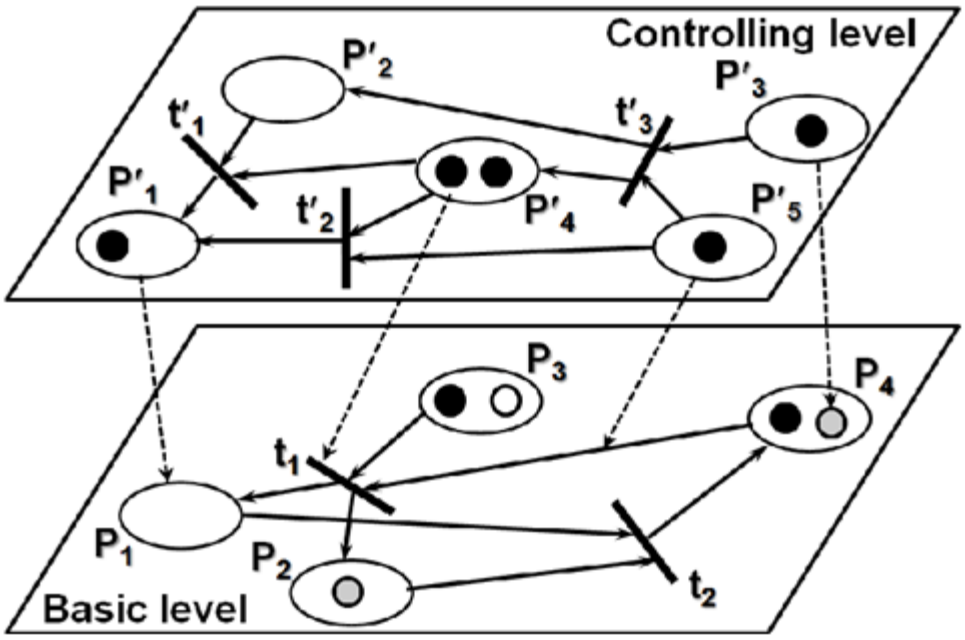
Zuohua Ding ; Yuan Zhou ; Mengchu Zhou



Ding, Z., Zhou, Y., & Zhou, M. (2016). Modeling Self-Adaptive Software Systems with Learning Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *46*(4), 483-498.

Traditional models unable to handle the behaviors that change at runtime in response to environmental context changes. An adaptive Petri net is an extension of hybrid Petri nets by embedding a neural network at some special transitions. The proposed net can make adaption decisions in run time.

MetaPetriNets: the time has come !

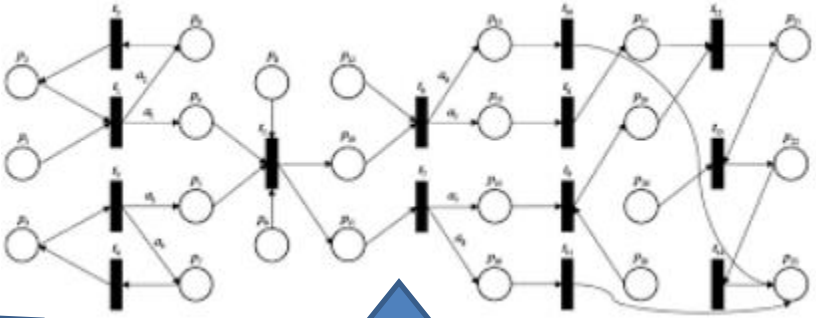


- *A **metapetrinet*** is able not only to change the marking of a petrinet but also to reconfigure dynamically its structure
- Each level of the new structure is an ordinary petrinet of some traditional type.
- A basic level petrinet simulates the process of some application.
- The second level, i.e. the metapetrinet, is used to simulate and help controlling the configuration change at the basic level.

Savolainen, V., Terziyan, V. (1999). Metapetrinets for Controlling Complex and Dynamic Processes. *International Journal of Information and Management Sciences*, *10*(1), 13-32.

Multilevel Petri Nets (MetaPetriNets) were proposed for facilitating flexible (smart and adaptive) modeling and control of complex dynamic processes. A MetaPetriNet is able not only to change the marking of a petri net but also to reconfigure dynamically its structure. Each level of the MetaPetriNet is an ordinary petri net. A basic level simulates the process of some application. The second level is used to simulate the context and help controlling the configuration change at the basic level. There is conformity between the places of the second level structure and places or transitions of the basic level structure. The main control rule is such that a certain place or transition is removed (locked) from the present configuration of the basic level if the corresponding place at the metalevel becomes empty. If at least one token appears to an empty metalevel place, then the originally defined corresponding basic level place or transition immediately is created back to the configuration (unlocked). We also introduce a multilayer MetaPetriNet where every upper layer may change the configuration of the previous lower layer. In such a case even a few layers with a simple structure are able to simulate quite complicated processes at the basic level. We assert that MetaPetriNets offer more compact facilities to the models of complex dynamic processes than single level petri nets.

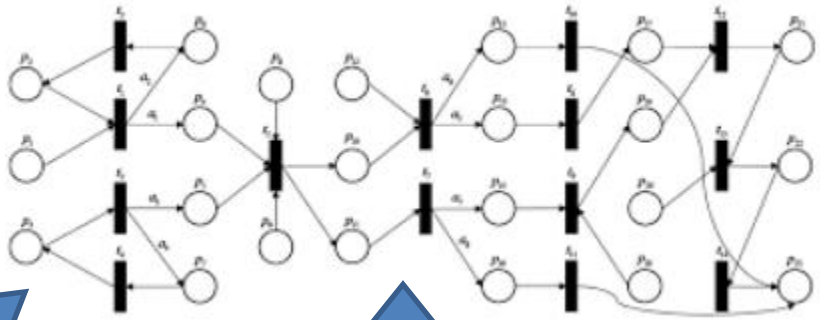
Simple scenario: human observes context change and updates the model configuration accordingly

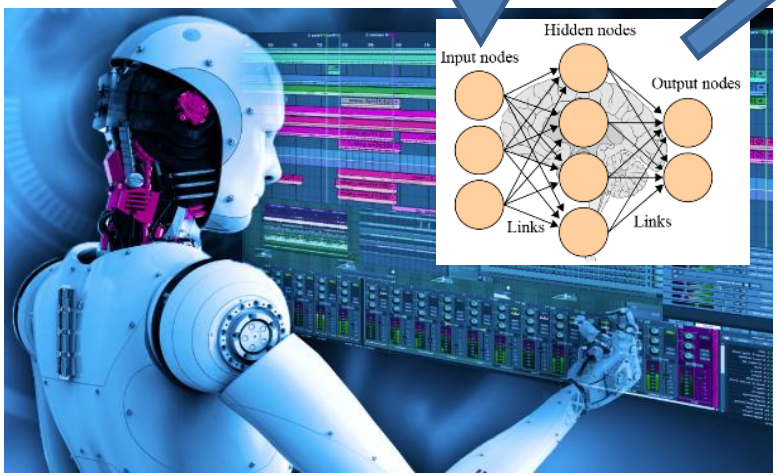


If we want a *self-adaptive* model;
also *smart* one (*capable to learn*);
and, finally, capable to *reconfigure itself autonomously* while observing and *addressing the context change*,...
we need to marry the concepts of *self-adaptive software systems*, *context-adaptive Petri Nets*, *MetaPetriNets*, Machine Learning (e.g., *Deep Neural Networks*) into one model:

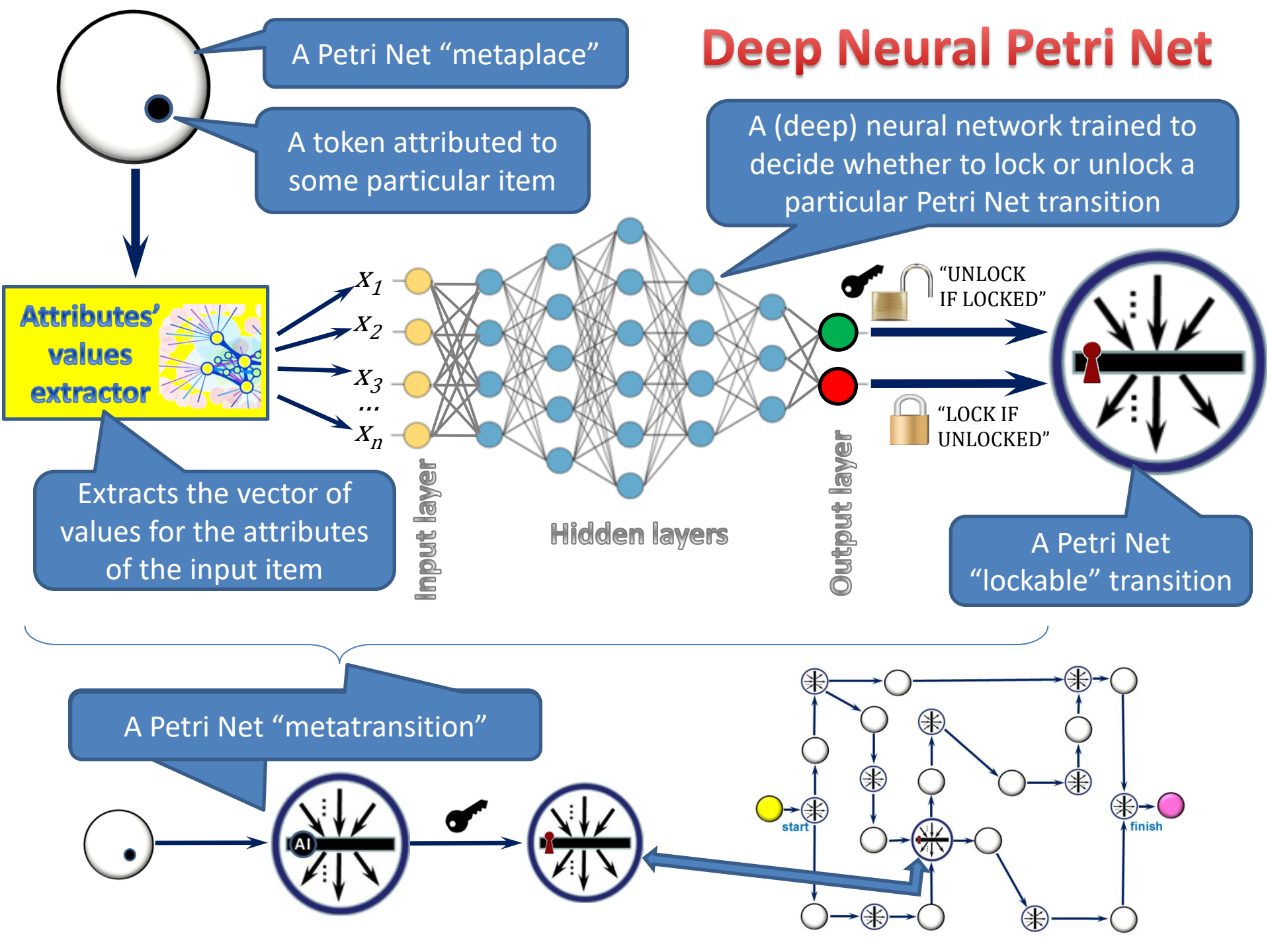
"Deep Neural Petri Net" (DNPN)

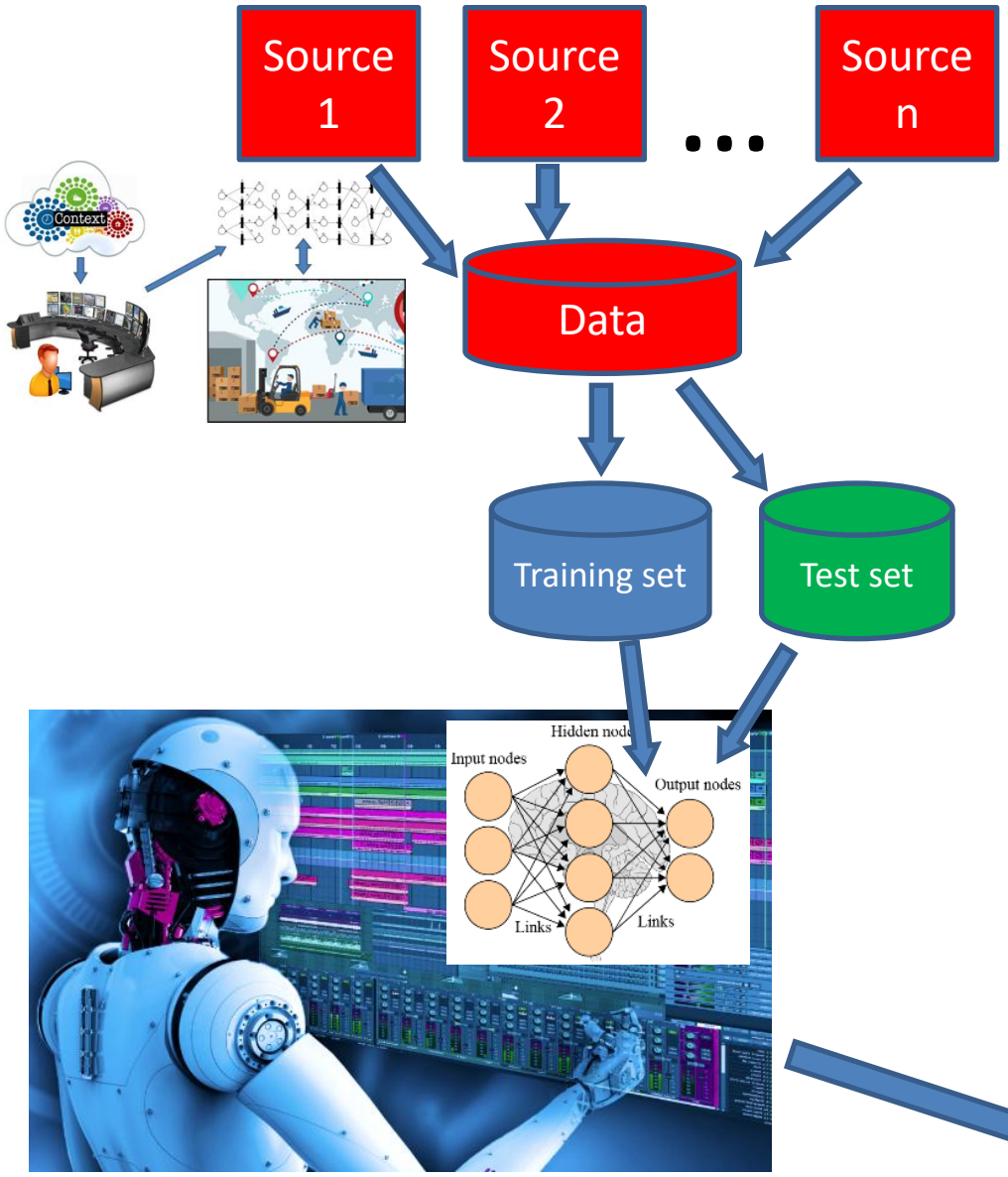
This is what we need





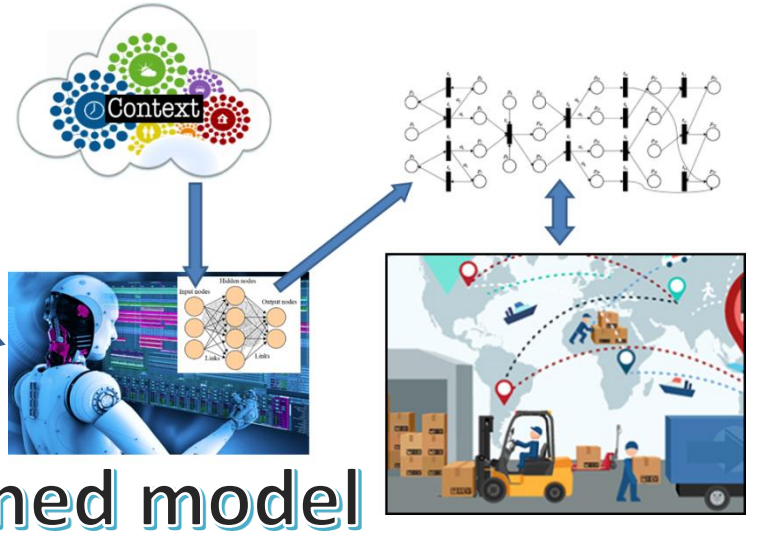
Deep Neural Petri Net



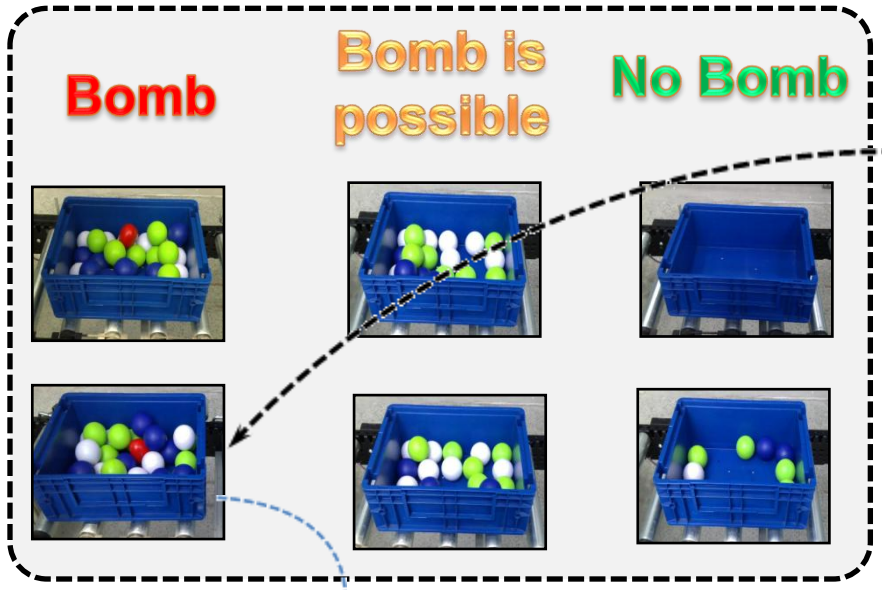


Intelligent component (i.e., Neural Network) of the model requires *training and testing* phase (Machine Learning). Training data is needed. Do we trust the sources of the training data?

Training and testing of the model



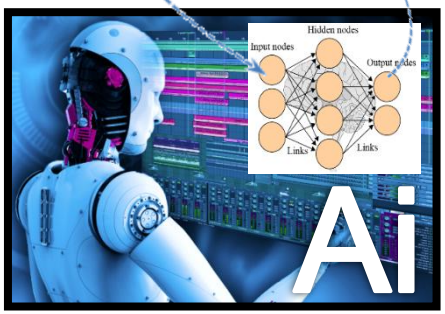
Trained model



Sample scenario

Who I am ?

You are bomb !



If bomb

If bomb is possible

If not a bomb