
On pi-Calculus and its Applications

Ing. Luca Gardelli
lgardelli@deis.unibo.it
lgardelli@unibo.it

DEIS - Dipartimento di Elettronica Informatica e Sistemistica
Università degli Studi di Bologna sede di Cesena
via Venezia 52, 47023, Cesena (FC) Italy
Ph: +39 0547 339244 Fax: +39 0547 339208

Outline

I. Pi-calculus

- Definition & syntax
- Extensions
- Tools

II. On the applicability of pi-calculus to self-org. Systems

- Definition of self-org. system & emergent property
 - Pros and Cons
 - The role of Spi in our research
 - Our case study
-

Seven **Myths** of Formal Methods

Hall 1990

- 1) Formal methods can guarantee perfect software and eliminate the need for testing
- 2) Formal methods are all about proving programs correct
- 3) Formal methods are only useful in safety-critical systems
- 4) Formal methods require highly trained mathematicians
- 5) Formal methods increase development costs
- 6) Formal methods are unacceptable to users
- 7) Formal methods are not used on real large-scale systems

Informal Definition

Milner 1992

Pi-calculus is a calculus for **communicating systems** in which one can naturally express processes which have changing structure.

Not only may the component agents of a system be arbitrarily linked, but a communication between neighbours may carry information which changes that linkage.

About Milner's definition of Agent

- In pi-calculus an agent is actually represented as a process
- Milner considers only agents with a finite behavior
- Milner gives no further details about agents, and usually the term agent is even omitted
- It also gives no details about the environment which has become one of the most important issues in MAS
- Hence in pi-calculus you can describe systems that can be conceived in terms of processes

History

- Pi-calculus is an extension of Milner's process algebra called **CCS** – Calculus for Communicating Systems
- It adds the feature of **process mobility**, i.e. processes having changing control structure, exploiting the results of Engberg and Nielsen
- CCS has been successfully applied for modelling several electronics and software systems

Syntax

Milner 1992 & 1993

$$P ::= 0 \mid P_1 + P_2 \mid \bar{y}x.P \mid y(x).P \mid \tau.P \mid P_1|P_2 \mid (\nu x)P \mid [x = y]P \mid !P$$

- 0 is the empty process, and it's called *inaction*: since it is very common it is often omitted;
- summation $P_1 + P_2$ means that the process can perform either P_1 or P_2 ;
- the prefix $\bar{y}x$ is a sort of output port, so $\bar{y}x.P$ means send x across y channel and then behave like P ;
- the prefix $y(x)$ is a sort of input port, so $y(x).P$ means receive a value across y channel, name it x and then behave like P ;
- the silent prefix τ means that a silent action is performed;
- the composition $P_1|P_2$ means that the two processes are executed in parallel
- the restriction $(\nu x)P$ means that the process behaves like P except for the fact that any action across x channel is prohibited;
- $[x = y]P$ means that the process behaves like P if y matches x , otherwise 0 ;
- the replication $!P$ means that you can have as many copies – but a finite number – as you wish, i.e. $P|P|P|\dots$

Reduction Rules

Milner 1992 / 1993

$$COMM : (\dots + x(y).P) \mid (\dots + \bar{x}z.Q) \rightarrow P\{z/y\} \mid Q$$

$$PAR : \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}$$

$$RES : \frac{P \rightarrow P'}{(\nu x)P \rightarrow (\nu x)P'}$$

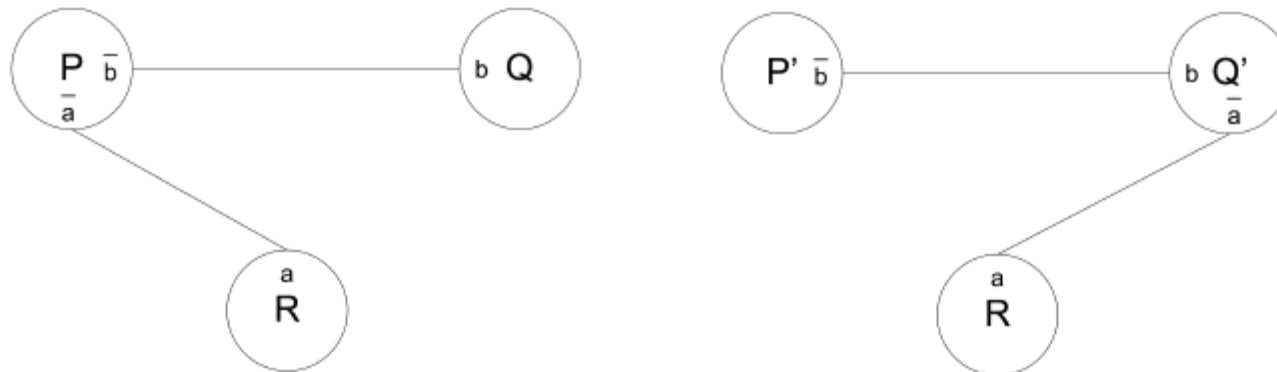
$$STRUCT : \frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$$

Name Passing Example Milner 1992

- 3 processes P,Q,R: P and R share channel a, P and Q share b
- P wants to send the value 42 to R
- P wants to delegate that task to Q

$$(\nu a)(\nu b)(\bar{b}a.\bar{b}42.P' \mid b(x).b(y).\bar{x}y.0 \mid a(z).R')$$

$$(\nu a)(\nu b)(P' \mid \bar{x}y.0 \mid a(z).R')$$



What can I do with Pi?

- Investigate fundamental questions in concurrency & communication
- Formal proofs for SMALL systems
- Write programs in a “functional style” using tools like Pict ... but it's better suited as a framework for higher-level languages
- It's a good **modeling language**: e.g. network and security protocols have been successfully modeled in pi-calculus
- It can be used also for simulations... but it lacks one important feature: **stochasticity**

Stochastic Extension to Pi

Priami 1995

- Choices in Pi are nondeterministic
- Priami introduced an extension to Pi in order to make choices probabilistic
- Each channel is associated to an *activity rate* r
- The usage of a resource is a random variable with exponential distribution defined by r
- Given $P.(P_1+P_2+\dots+P_n)$ the probability of the transition between P and P_i is

$$p_i = \frac{r_i}{\sum_{j=1..n} r_j}, \quad 1 \leq i \leq n$$

...and now?

- Once we have modeled our system with stochastic pi-calculus maybe we want to **simulate** its working for
 - Feasibility of the whole system
 - Assessment of global properties
 - Performance analysis
 - ...
- For that purpose we need an interpreter of Spi specifications

Stochastic Pi Machine (SPiM)

Phillips

- The Stochastic Pi Machine (SPiM) is a simulator for the stochastic pi-calculus that can be used to simulate models of **Biological systems**.
- The machine has been formally specified, and the specification has been proved correct with respect to the calculus.
- It is a quite fast and optimized simulator
- The syntax is quite different from the one of pi-calculus and have been added more features

SpiM: KNa2Cl example

```
(* K + Na + 2Cl <==> K+ + Na+ + 2Cl- *)
directive sample 0.03
directive plot Cl(); Na_plus(); K_plus()

new ionize1@100.0:chan
new deionize1@10.0:chan
new ionize2@30.0:chan
new deionize2@20.0:chan

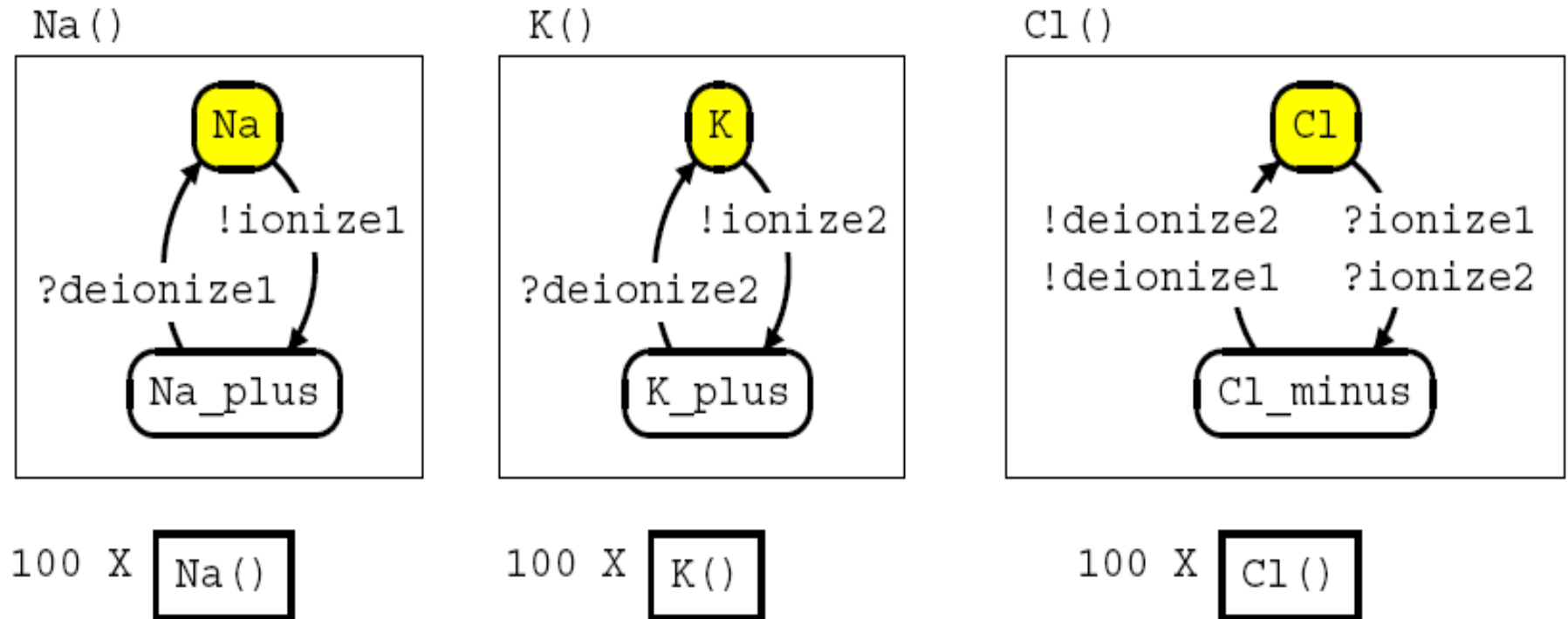
let Na() = !ionize1; Na_plus()
and Na_plus() = ?deionize1; Na()

let K() = !ionize2; K_plus()
and K_plus() = ?deionize2; K()

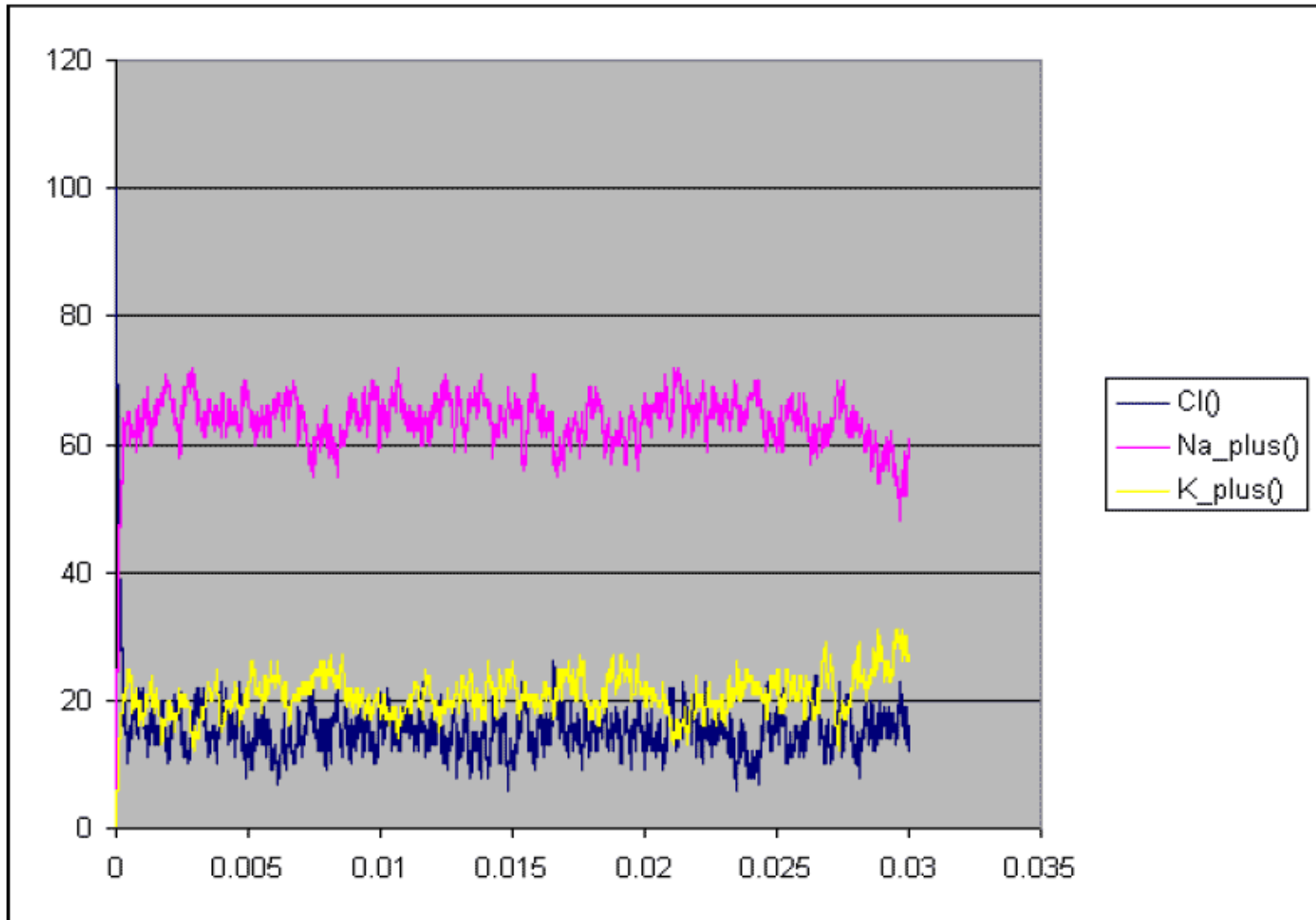
let Cl() =
  do ?ionize1; Cl_minus()
  or ?ionize2; Cl_minus()
and Cl_minus() =
  do !deionize1; Cl()
  or !deionize2; Cl()

run 100 of (Na() | Cl() | K())
```

SpiM: KNa2Cl example



SpiM: KNa2Cl example



Comments

- It is very simple to model systems similar to KNa_2Cl
- The mapping is quite easy when the essence is about signaling for inhibition or catalysis
- From a scientific it is very interesting to have such tool, but how it can usefully applied to MAS?

Definition of Self-Organization

A system is said to be self-organizing if it's able to re-organize itself upon environment changes autonomously via local interactions between its parts.

Definition of Emergent Property

A system property is said to be emergent if it cannot be described using the same - or equivalent - ontology used to describe the system itself.

On the applicability of Spi-calculus to self-org. systems

- Spi-calculus is not directly/strictly related to SOS
- Why we started considering Spi for SOS?
 - it is well suited for large scale systems with weak heterogeneity
 - it is a modeling language for communication / interaction
 - full compositionality
 - it has already been applied to biology/chemistry research, e.g. see Cardelli's works
 - it features stochasticity

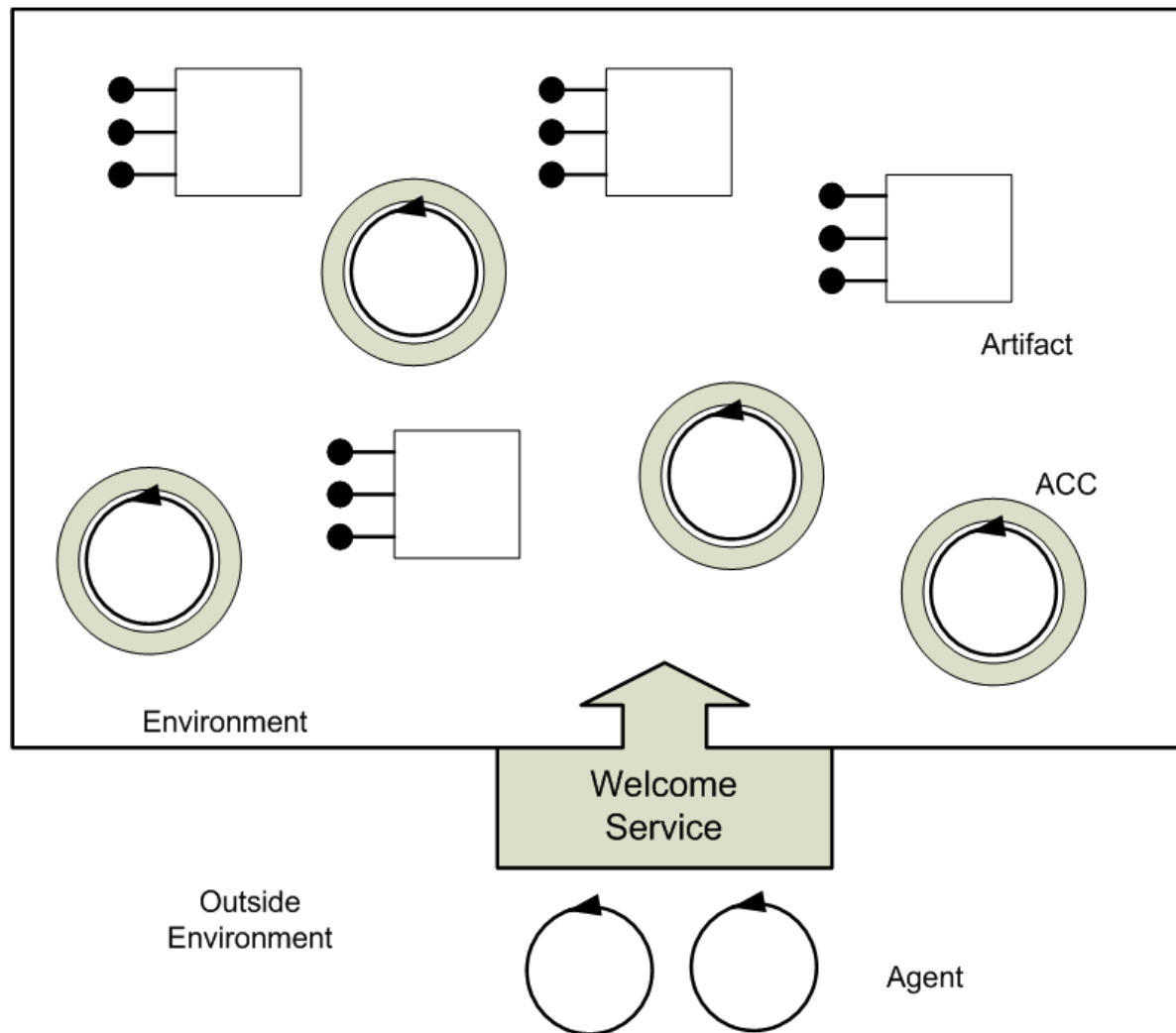
Limitations

- Since it is quite young there are many issues that should be tackled
- e.g. it does not take into account *explicitly* locality issues: this might be an interesting point of extension
- Feasible models of complex heterogeneous systems can be very abstract / high-level

Role of Spi in our research

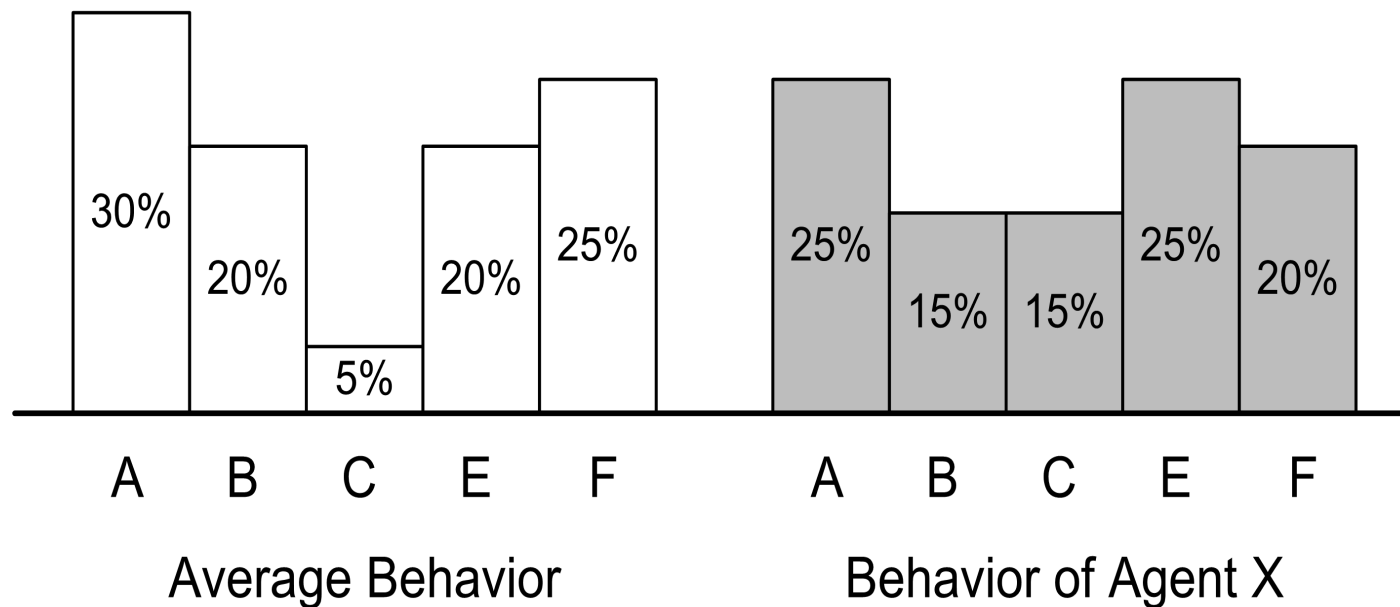
- 1) Feasibility study at early stages of analysis
- 2) “Observe” global system properties actually happening & detect emergent properties
- 3) Coarse tuning system parameters
- 4) Performance evaluation, e.g. PEPA

Our case study



Our case study

- We can detect abnormal behaving agents through analysis of the distribution of their actions
- If an agent X is behaving “differently” from the average – especially for critical actions – we may decide to further inspect him or deny access to resources



Our case study

