Rule-Based Support for Integrated Security Systems

Balázs Arthofer **Gergely Vakulya** **Gyula Simon**

Department of Computer Science
Faculty of Information Technology
University of Pannonia
Hungary

24. 06. 2009.
MENDEL - 15th International Conference on Soft Computing

Table of contents

Introduction

Security threats

Physical

- Fire
- Power loss
- Illegal enter
- ...

Logical

- Viruses
- Data loss
- Unauthorized login
- ...

Human

- Industrial spy
- Deliberate damage
- ...

Traditional security systems

- Independent subsystems
- Handwritten rules

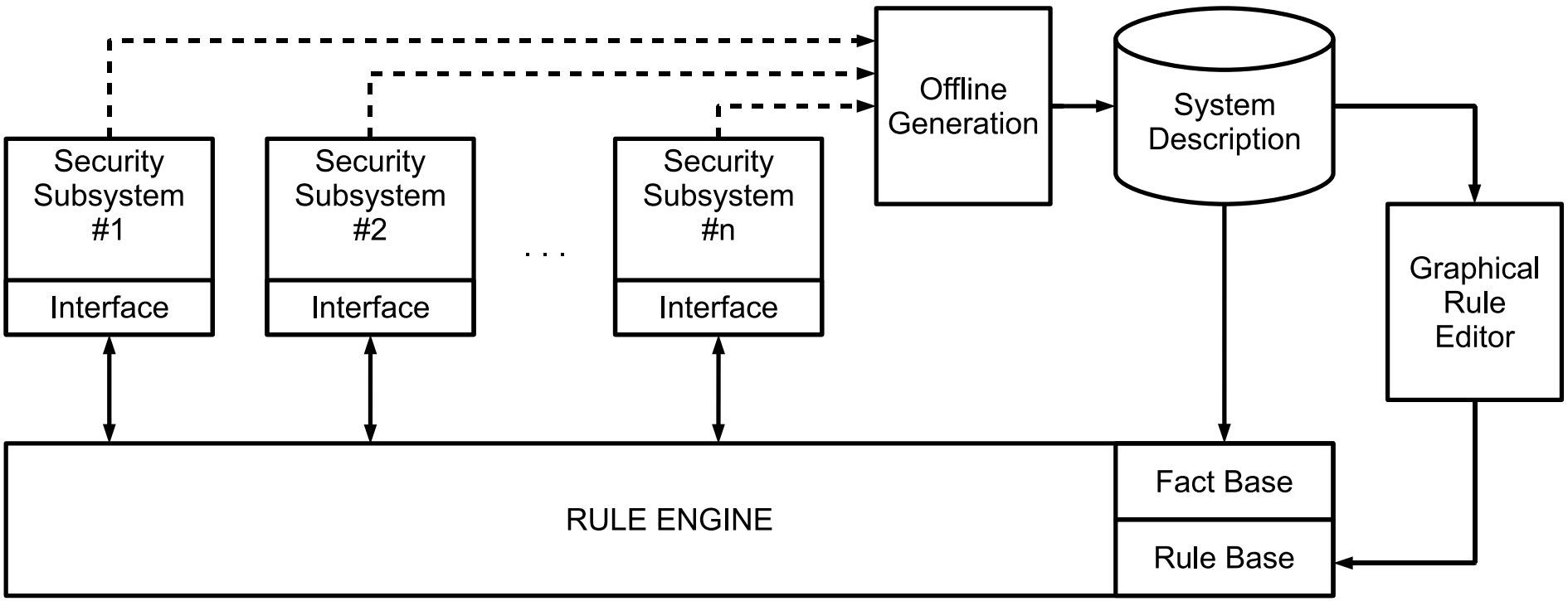
Integrated security systems

- Utilization of coherences
- Graphical rules editing
- Rule generation

Integrated Security Systems (ISS)

Security subsystems

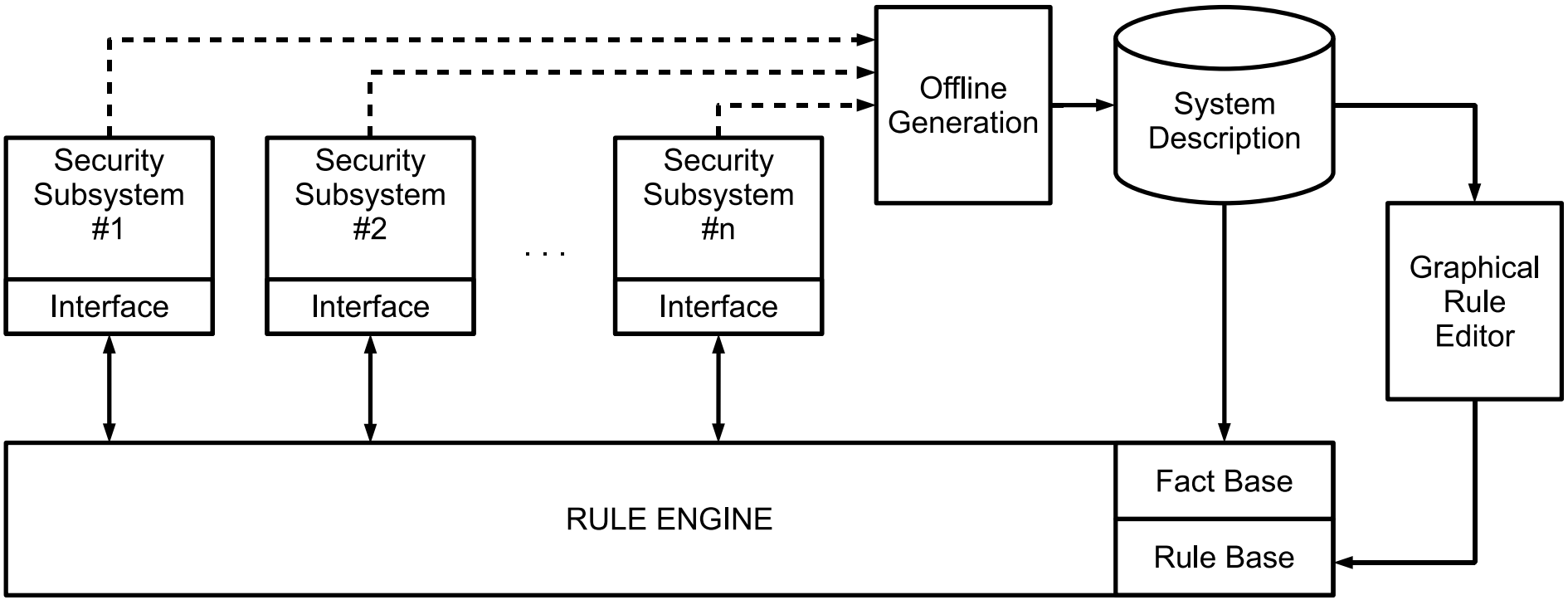
- The original subsystems are kept
- The sensors are accessed through an additional interface
- The subsystems send measures from the sensors regularly



Integrated Security Systems (ISS)

System Description Database

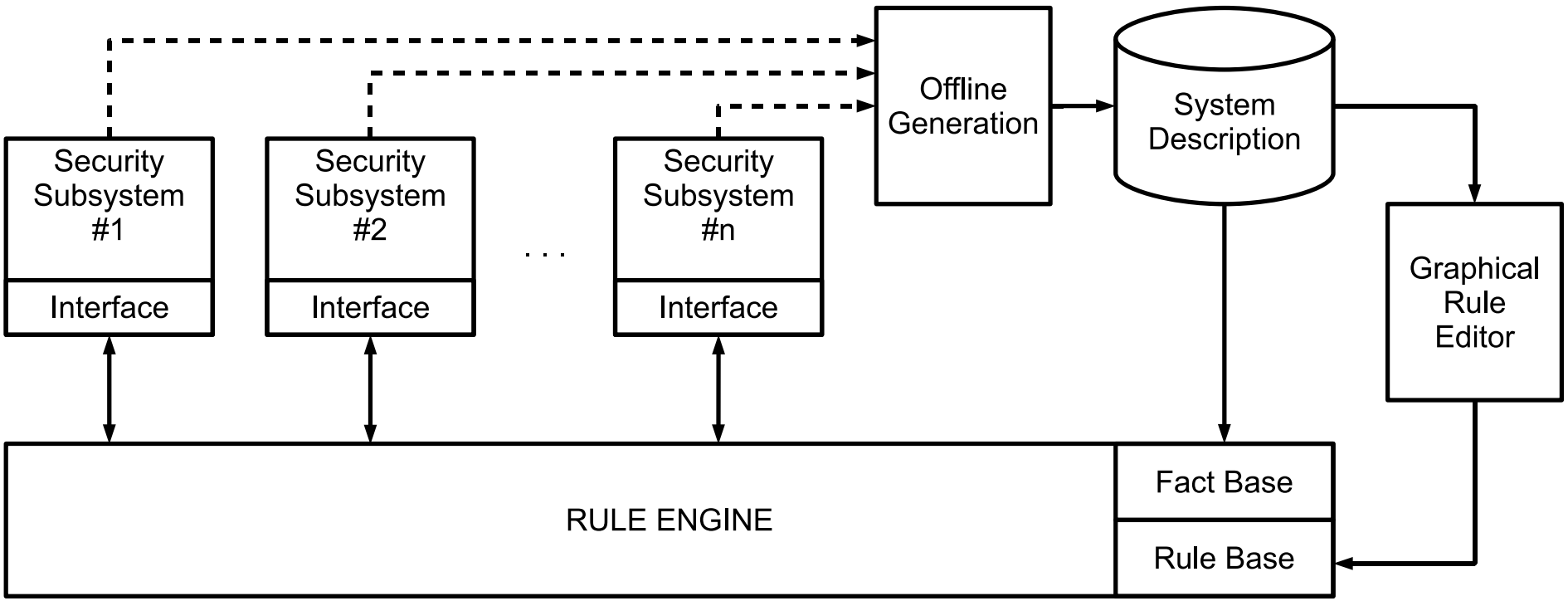
- Stores information about all subsystems' sensors
- Offline generation of data form security subsystems



Integrated Security Systems (ISS)

Graphical Rule Editor

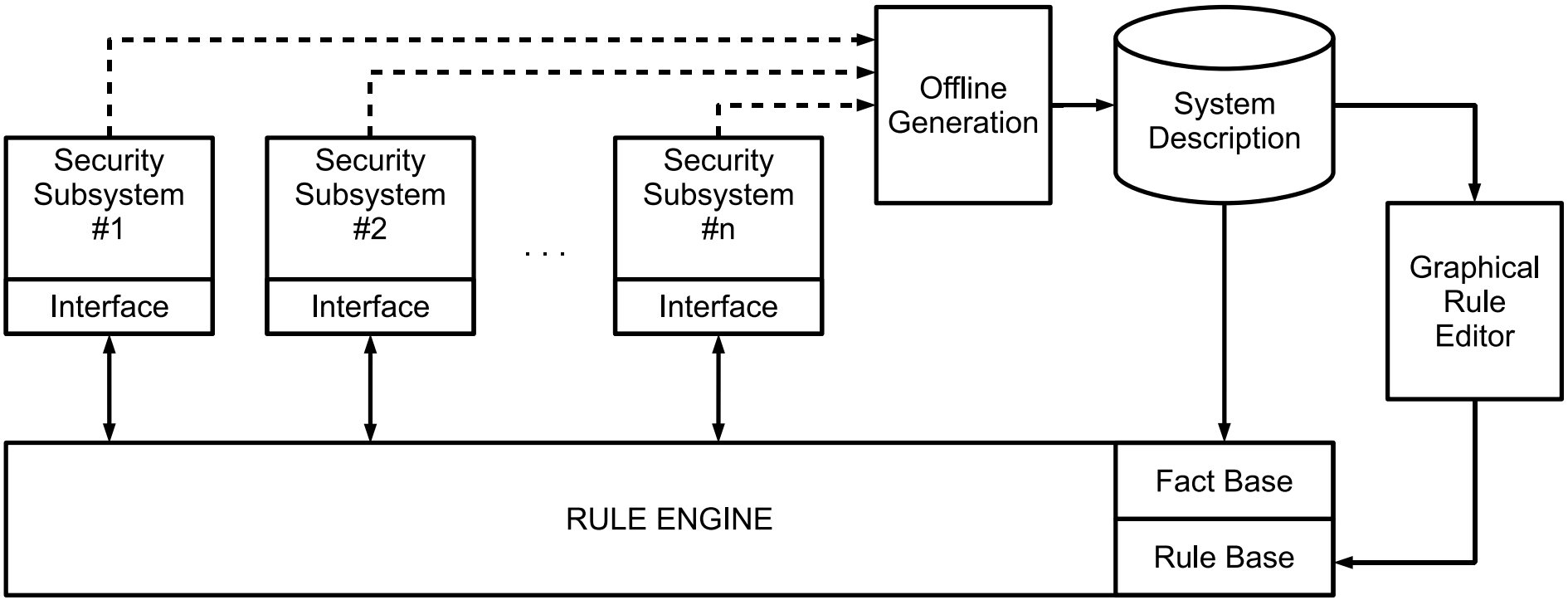
- Making rules is possible for people without deep knowledge about the system internals
- Rules are generated from the graphical representation



Integrated Security Systems (ISS)

Rule Engine

- Decision can be made using rules
- If it is necessary, actuations can be made (warnings, alarms, . . .)



Rule based support for ISS

System Description Facts

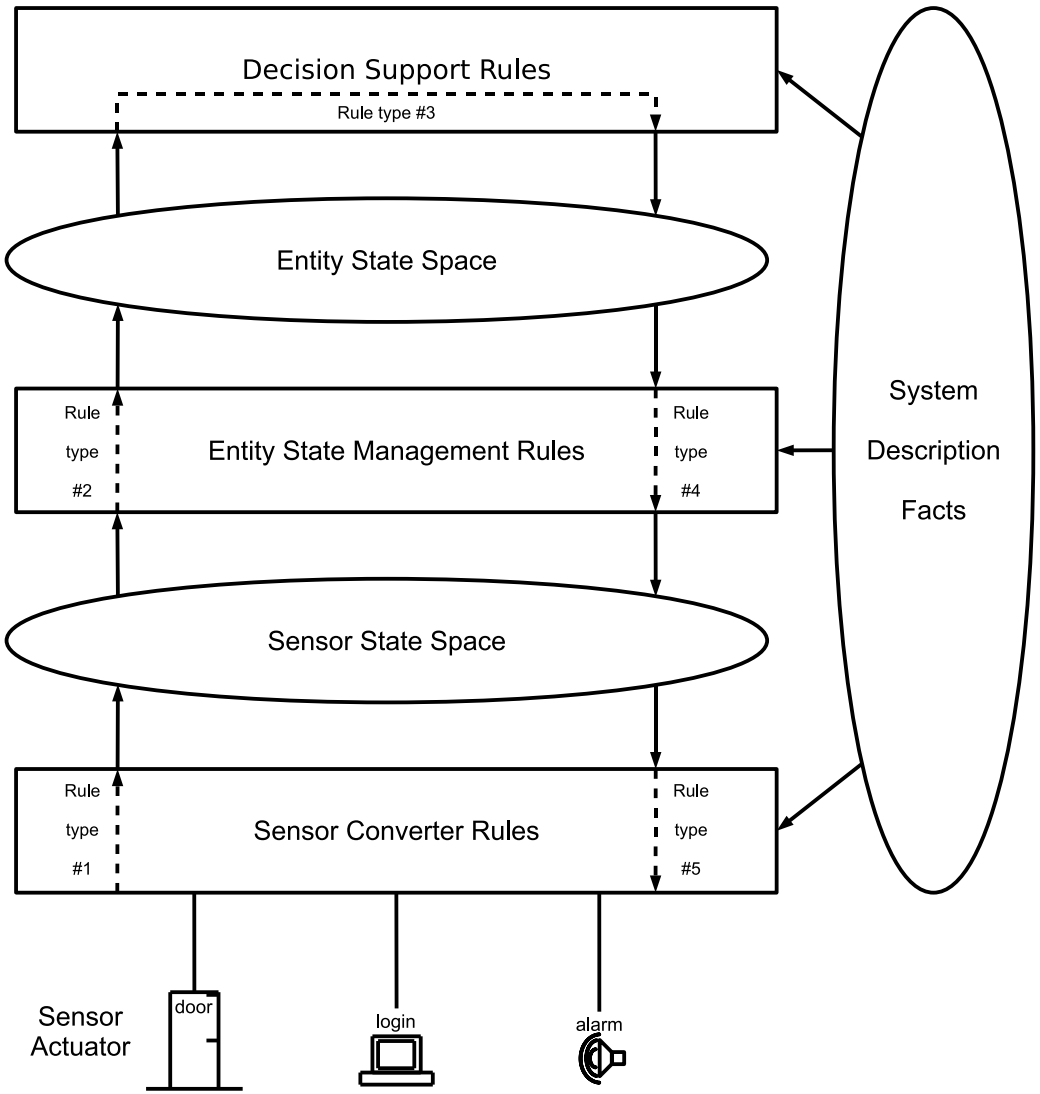
- Sensor locations
- Computer locations
- Employee's data

Sensor State Space

- Sensor standardization
- Sensor's current state

Entity State Space

- Entity abstractization
- Entity's current state



Rule based support for ISS

Rule type #1

Measures of the sensors are converted to SSS

Rule type #2

States of the sensors are converted to ESS

Rule type #3

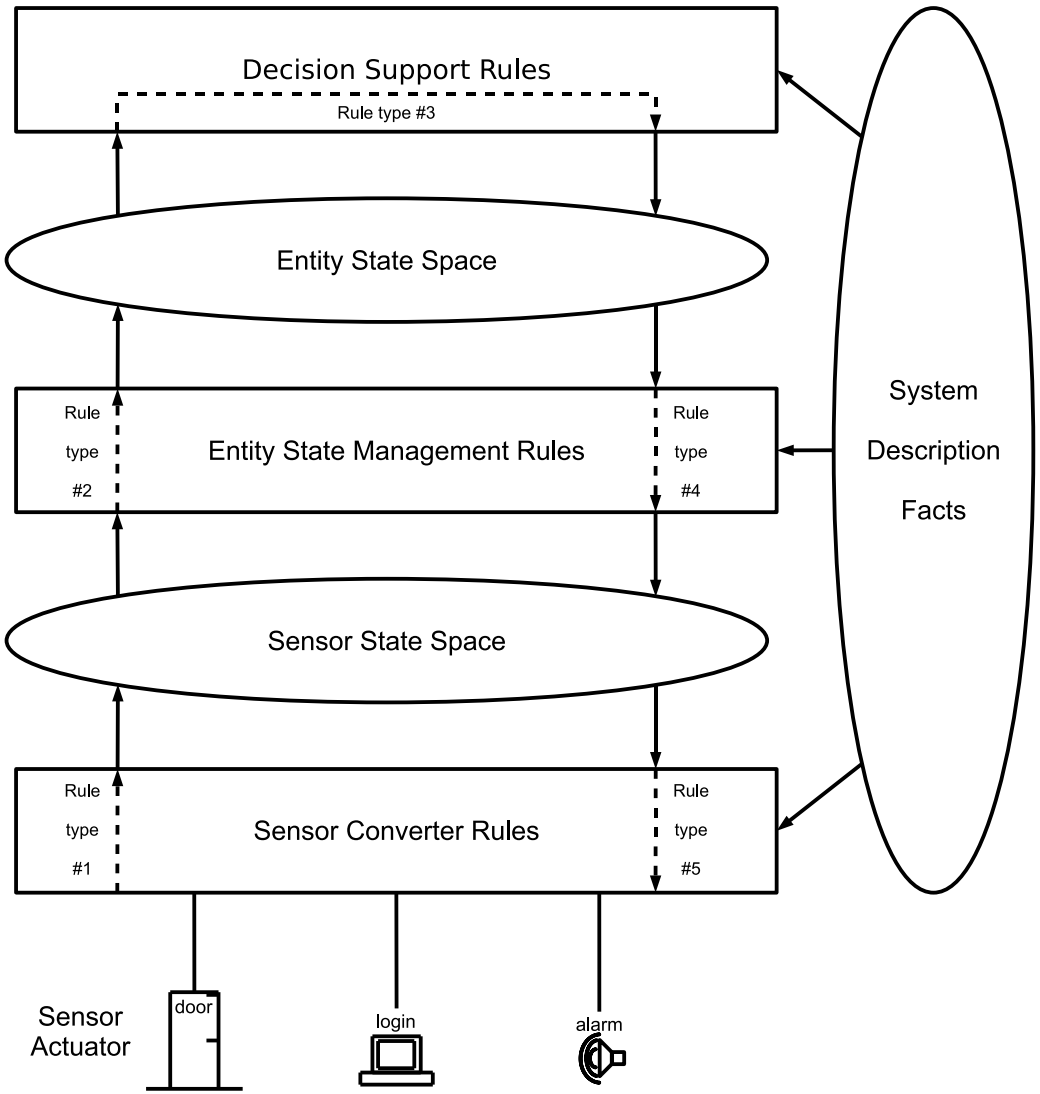
Decision and actuating

Rule type #4

Actuator selection

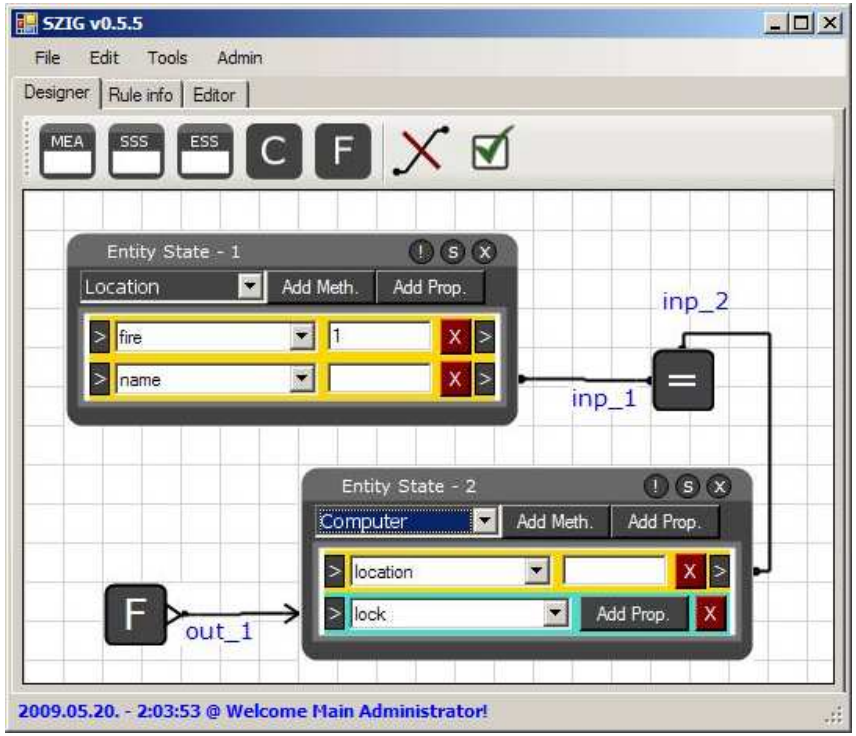
Rule type #5

Actuating is converted to actuator's language



Example

- There is a location, where fire is signaled,
- there is a computer at that location,
- lock that computer



JESS rule

```
(defrule firelock "Fire locks computers"
?fact1 <- (ESS_computer (name ?var1) (location ?var2))
?fact2 <- (ESS_location (fire "1") (name ?var2))
=>
(assert (ESS_computer.lock (now (?functions getCurrentTime)) (target ?var1))))
```

Summary

- Weakness of classical security systems is recognized
- An architecture is proposed using integrated approach
- The rule based system is implemented and interfaced to a commercial security system
- A rule editor is written to ease the rule-creating process
- A rule converter backend is written to support the rule editor
- The complete system is successfully tested in real environment

Thank you for your attention!