

# Radio Frequency IDentification: Evaluation of the Technology Supporting the Development of an Assets Tracking Application

**Dominique Guinard**

Bachelor Thesis in Computer Science

Department of Computer Science

University of Fribourg, Switzerland

September 2005.



# Content

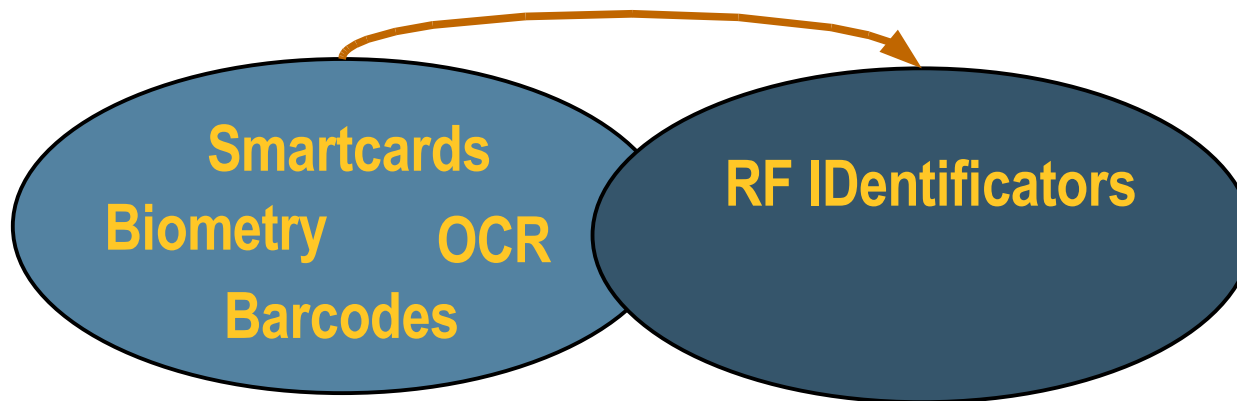
- **Introduction to the RFID and EPC standards**
- **Software architecture**
- **Hardware settings**
- **Conclusion**

# Content

- **Introduction to the RFID and EPC standards**
- Software architecture
- Hardware settings
- Conclusion

# Introduction

## About Auto-ID Technologies

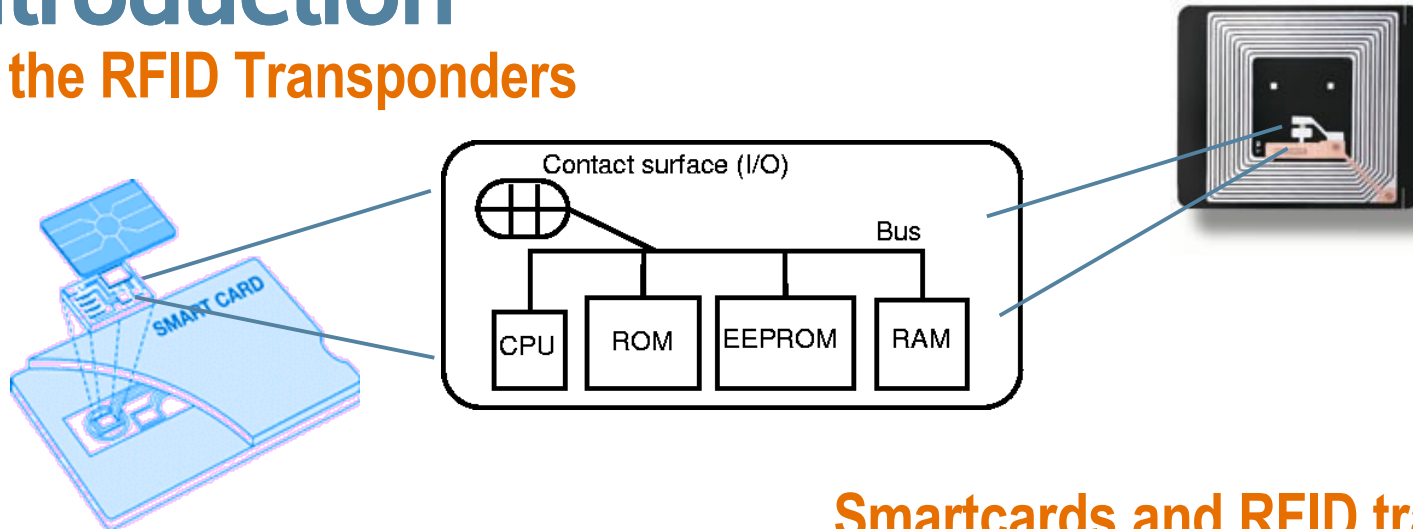


## Auto-ID technologies

- A simple idea drives all the Auto-ID technologies:  
Enabling computers to **identify** assets, people or animals.
- The RFID as an evolution of the Auto-ID technologies.
- The RFID adds the notion of **contact-less** identification.
- It also enables:
  - The reading of identifiers that are **in motion**.
  - The detection of objects that are **not in line of sight**.

# Introduction

## To the RFID Transponders

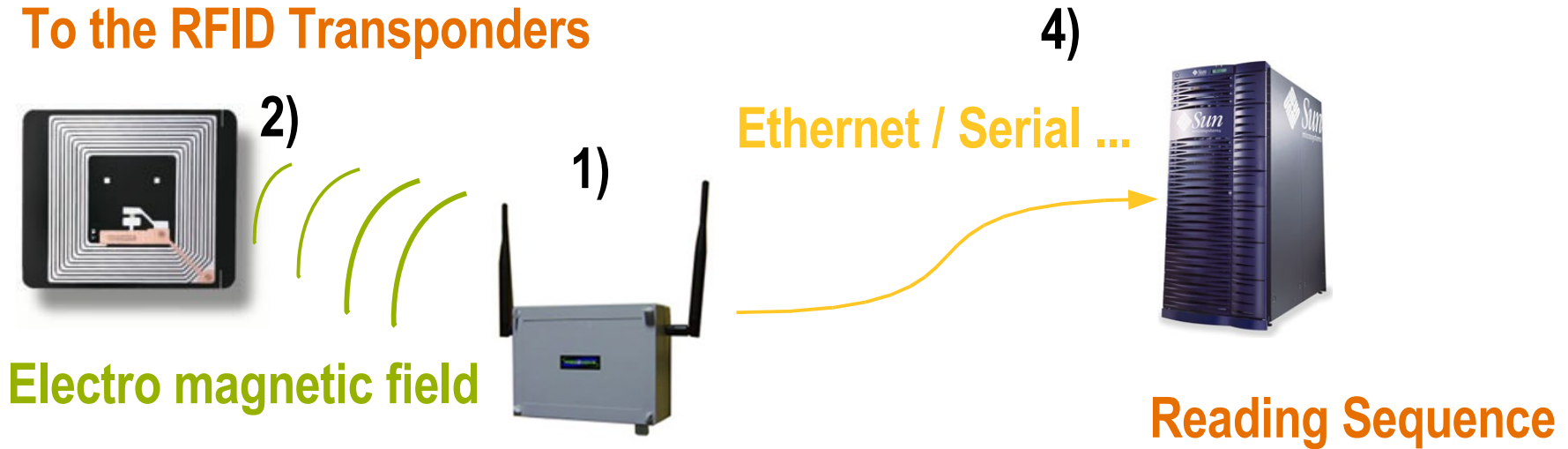


## Smartcards and RFID transponders

- RFID transponders (aka tags) are often **presented** as a new generation of **barcodes**. However, technically speaking the tags are **more related to smart cards**.
- Many kinds of transponders are available:
  - **Active**: with internal power supply.
  - **Passive**: externally powered.
  - **Readable**: cheapest kind of transponders.
  - **Readable/Writable**: can persist a small (~nn Kbytes) amount of data.
  - **Activation frequency**: LF (KHz), HF (MHz), UHF (MHz), microwaves (GHz).

# Introduction

## To the RFID Transponders



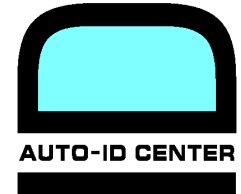
- 1) The RFID reader (aka sensor) emits an **electro magnetic field**.
- 2) The transponder **converts** the field into a source of power.
- 3) The tag is powered, the sensor can now start reading(/writing) the content emitted by the tag.
- 4) The RFID events are transmitted to a computer in charge of processing them.



# Introduction

## To the EPC Standards: the Physical Markup Language

EPCglobal 



[...]

```
<Sensor xmlns="urn:autoid:specification:interchange:PMLCore:xml:schema:1">
  <ns1:ID xmlns:ns1="urn:autoid:specification:universal:Identifier:xml:schema:1">
    urn:epc:id:gid:1.255.1
  </ns1:ID>
  <Observation>
    <DateTime>2004-12-28T13:46:52.292+01:00</DateTime>
    <Tag>
      <ns4:ID xmlns:ns4="urn:autoid:specification:universal:Identifier:xml:schema:1">
        urn:epc:id:gid:1.1.102
      </ns4:ID>
    </Tag>
```

[...]

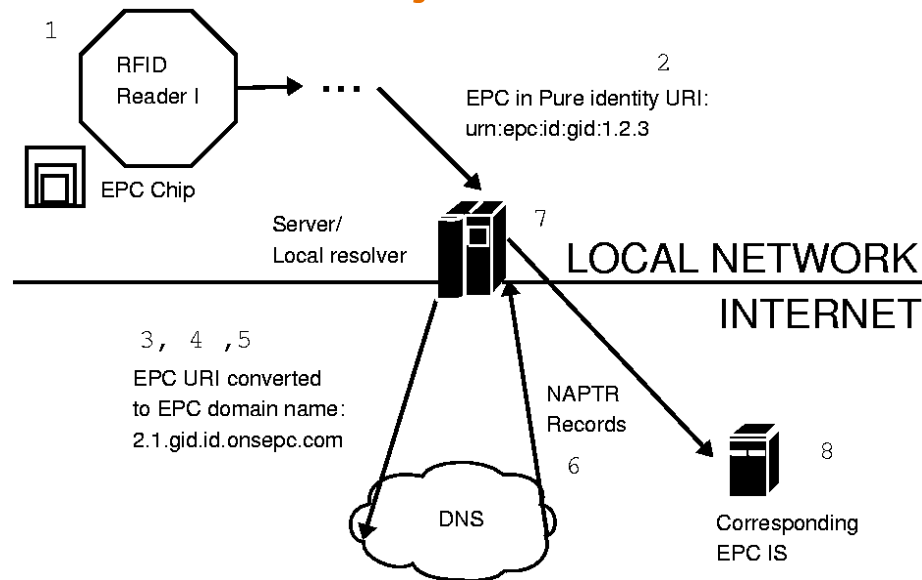
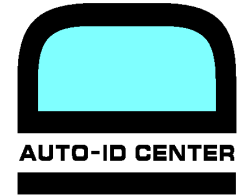
A PML extract

- The PML provides a standardized format for **interchanging the RFID events** (or observations) within an EPC Network.
- Based on **XML** and standardized by two schemata.
- Can be read by **both humans and computers**.

# Introduction

## To the EPC Standards: the Object Name Service

EPCglobal 



A typical  
ONS query

- A simple idea: having the EPC of an object **where can I retrieve (authoritative) data** about it ?
- Designed on top of the **DNS** (Domain Name Service).
- Root directory of ONS managed by VeriSign.
- First ONS standard still not ratified.

# Introduction

## To the EPC Standards: towards an Internet of Things



## The EPC standards in action

- Put all together, the EPC standards **converge towards** a global network of the physical objects surrounding us: the **EPC Network**.
- The EPC Network is **still young** but its potential might well boost the number of researchers and manufacturers around the technology.

# Content

- Introduction to the RFID and EPC standards
- **Software architecture**
- Hardware settings
- Conclusion

# Software Architecture

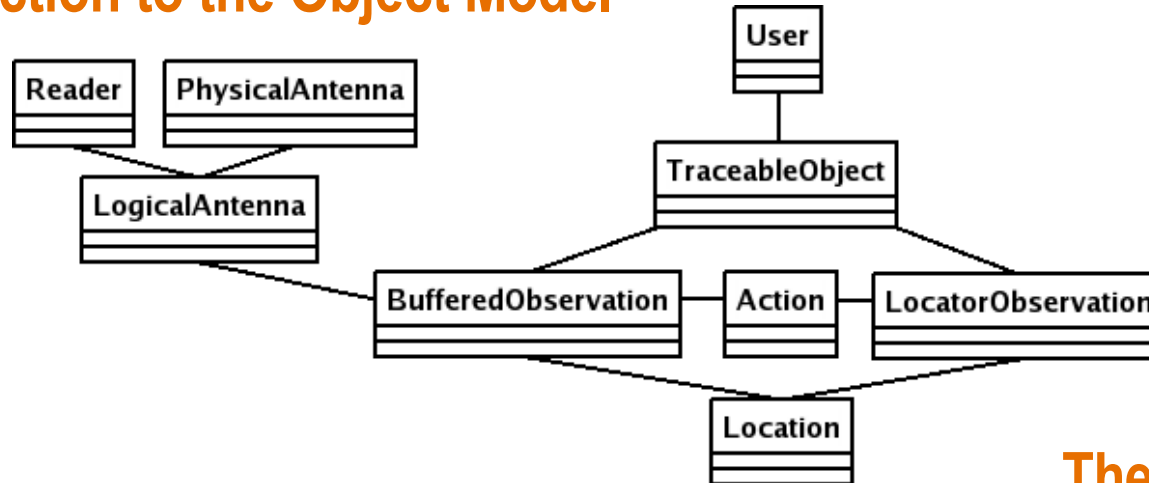
## The RFIDLocator: Use Cases and Definition



- The RFIDLocator is an application for **tracking assets within a predefined area** (typically a set of buildings).
- Two use cases examples:
  - The Data Center:
    - One RFID transponder on each server
    - Readers at the entrance of each room
    - Installing and configuring the RFIDLocator
    - => using the application one can **find in what room** (or even rack) a particular machine is located.
  - The Hospital:
    - One RFID transponder on each medical file
    - Standalone and handheld readers set at central points
    - Installing and configuring the RFIDLocator
    - => using the application the **medical files can be traced and located** efficiently.

# Software Architecture

## Introduction to the Object Model



## The Object Model

- Reader: models an RFID reader (aka Sensor).
- PhysicalAntenna: a hardware component able to capture RFID events.
- LogicalAntenna: **Groups** 1..n PhysicalAntennae.
- LocatorObservation and BufferedObservation: **results** of an RFID event.
- TraceableObject: **models assets** equipped of an RFID tag and traced by the application.
- Location: a “place” within the area controlled by the RFIDLocator.
- Action: action assigned to the RFID events: **either IN or OUT** of the Location.

# Software Architecture

## The Services and Solvers

- The RFIDLocator is composed of **various services** called the Managers. There is basically one Manager per class of the Object Model:
  - TraceableObjectManager: used to return the history of observations made for a particular asset.
  - ReaderManager: used to inform the RFIDLocator about the “geographical” location of each RFID reader.
  - ObservationManager: used to persist an RFID event as an observation of the RFIDLocator (i.e. as a LocatorObservation).
  - PMLSimulator: used to generate RFID events for testing the application without the use of a concrete RFID reader.
- The Solvers are the **algorithms of the application**. These are used to decide whether an RFID event should be persisted as a LocatorObservation.
- The SensorListenerMessageDrivenBean is the **integration point** between the RFID readers and the RFIDLocator.

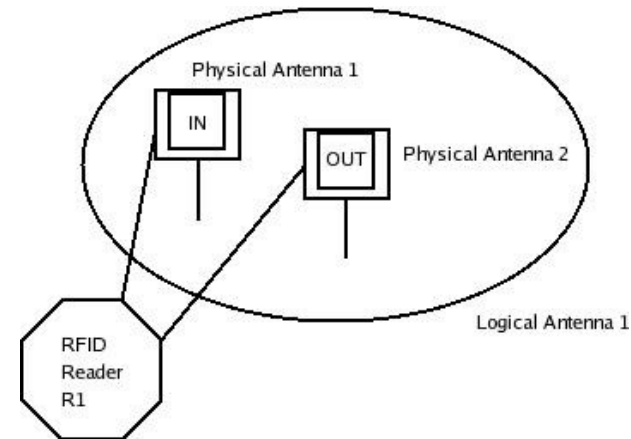
# Software Architecture

## Capturing the Direction of the Motion

- **How to capture the direction** of the motion without the use of special readers ?
- **By:**
  - Assigning a **single action**: either **IN** or **OUT** to each PhysicalAntenna .
  - **Grouping** 1..n PhysicalAntennae into a LogicalAntenna that aggregates multiple **RFID events (caught at the level of the PhysicalAntennae) into a single LocatorObservation.**
  - Assigning a **single Location**: e.g. “Room 91” or “Building 12” to each LogicalAntenna .
  - Deciding the action (**IN** or **OUT**) a LocatorObservation is bound to. This is done by using **modular algorithms** extended from a Solver class.

# Software Architecture

## Capturing the Direction of the Motion: Concrete Example

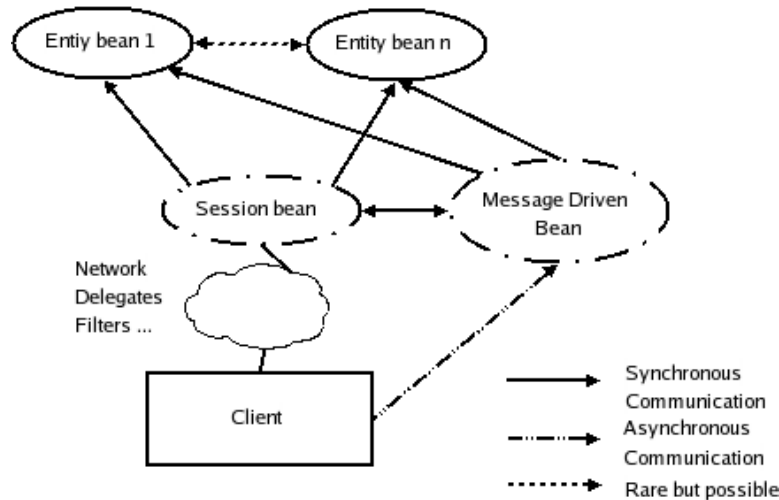


## Capturing the direction

- One RFID reader without genuine capabilities of capturing the direction of the motion.
- Two PhysicalAntennae, reporting IN respectively OUT RFID observations.
- One LogicalAntenna aggregating both PhysicalAntennae.
- The LogicalAntenna reports LocatorObservations attached to **either IN or OUT actions** for for “**room RM.3-113**”.
- This **decision relies** on the Solver (= algorithm) of the LogicalAntenna.

# Software Architecture

## Software Choices



## EJB Components

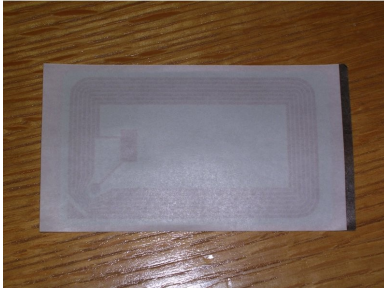
- The software components are developed using the **EJB (2.0) specification**.
- Where:
  - The Objects of the main Model are **Entity Beans**.
  - The Services and Solvers are **Session Beans**.
  - The integration point is a **Message Driven Beans** receiving **JMS** messages.
- The application is **deployed** on the Sun Java System Application Server (8.1).
- The **EPC middleware** is the Sun Java System RFID Software (Event Manager) version 1.

# Content

- Introduction to the RFID and EPC standards
- Software architecture
- **Hardware settings**
- Conclusion

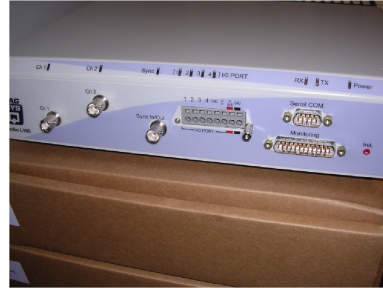
# Hardware Settings

## Reader, Tags and Converter



Tags

EPC Philips RAFSEC



RFID Reader

Medio L100



Converter

Moxa Nport Express

- In order to test the RFID technology a sample of the hardware **enrolled in an industrial application** is set.
- Transponders: 100 EPC Philips RAFSEC. An EPC of type GID is encoded on them.
- Reader: Medio L100 from Tagsys with a Serial output port.
- Converter: In order to propagate on the network the events recorded by the reader, a Serial to Ethernet converter from Moxa is used.

# Hardware Settings

## Computers

Fedora™  
PROJECT

IBM  
ThinkPad R40



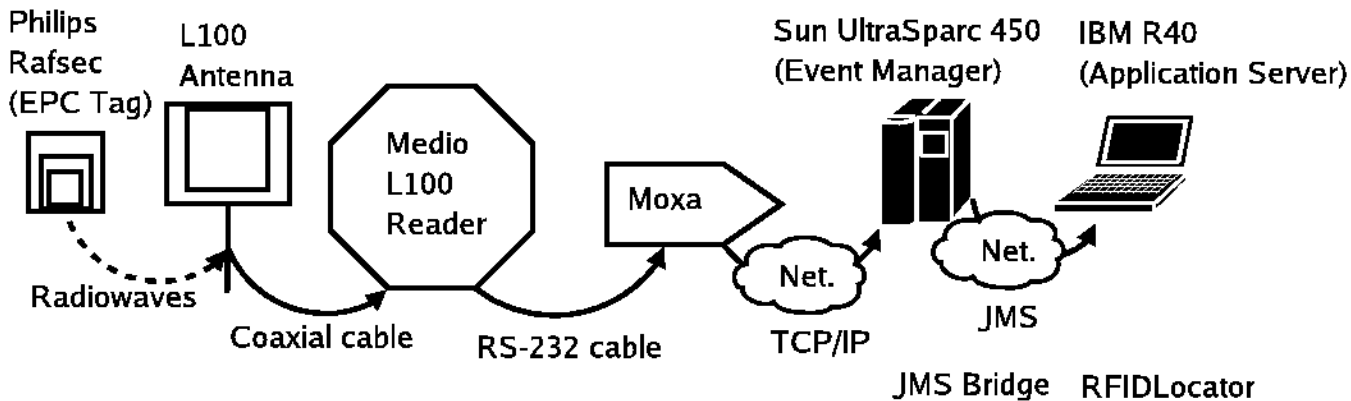
SUN  
Ultra Enterprise 450

solaris™

- The project was **developed** on an **IBM ThinkPad R40** (1.3 Mhz, Pentium M). The RFIDLocator is also installed on this machine (deployed on an Application Server).
- The **EPC Middleware** (Sun Java System RFID Software Event Manager) is installed on a **Ultra Enterprise 450** (4x300 Mhz, UltraSparc II), provided by Sun Microsystems.

# Hardware Settings

## Communication Between the Devices



## Demonstration Settings

- The output of the Medio L100 is propagated on the network using a Moxa converter.
- The UltraSparc 450 is in charge of **managing the Medio L100 and sending the events** to a JMS queue. For these tasks, it uses the EPC Event Manager provided by Sun Microsystems.
- The final application is deployed to an Application Server installed on a IBM R40. There, the **RFID events** are going to be processed and **converted into business events**.

# Content

- Introduction to the RFID and EPC standards
- Software architecture
- Hardware settings
- **Conclusion**

# Conclusion

## Open Problems



- The projects involves a **number of hardware components** interacting together. Making them communicate is not always trivial.
- The chosen **reader is not really adapted**. Its performances for reading tags in motion are quite disappointing.
- The **EPC network is a new** and fast evolving concept. Thus, some of the standards were redefined or even abandoned during the year.
- The of use of the **EJB 2.x specification somehow limits the object oriented paradigm** (limited inheritance and extensibility, etc.). This prevented us from designing a completely modular architecture for the solvers.
- EJBs permit to build “strong” distributed applications and are especially good for distributed transactions. However, due to their relative complexity the EJBs also significantly **complicate the debugging** process of the software.

# Conclusion

## Further Extensions



- The RFIDLocator is a **working prototype**. Not a fully-featured application.
- Making it available for commercial purposes would require:
  - A thinner granularity for the user management (roles, restricting access to the TraceableObjects, etc.).
  - A more **complete GUI** (modifying the readers' configuration, etc.).
  - A **cleaner front-end** (using patterns such as the Intercepting filter or frameworks like Struts/JSF) or the integration of RFIDLocator's core to an existing business system (like an ERP).
  - Detailed **load tests** to confirm the use of EJBs and the capability of the system (i.e. the Event Manager + RFIDLocator) to handle hundreds of events per second.

# Thank you for your attention...

- For the binaries, sources and the complete Bachelor Thesis please check:

[www.gmipsoft.com/rfid](http://www.gmipsoft.com/rfid)

- For more information about the Software Engineering Group check:

[diuf.unifr.ch/softeng](http://diuf.unifr.ch/softeng)

