Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

Using Java™ EE 5 in implementation of National EHR System

S.B.Stakic, D.Ljubojevic

Advanced Systems Guild LLC

www.AdvancedSystemsGuild.com

Purpose of the presentation

- Illustrate our experience of developing National Electronic Health Record System.
- Explain how we used Java EE 5 features in reaching several concurrent goals:
  - High security and data privacy protection,
  - International standards compliance,
  - Integration with numerous vendors that have implementations in different technologies, and
  - Testing and integration procedures.

Agenda

- Definitions of EMR, EHR, and the difference of EMR and EHR
- EHR systems features
- Standards applied
- Usage scenarios
- Mapping of Java EE 5 features to EHR-S mechanisms
  - Interceptors
  - Dependency injection
  - Persistence framework
  - Web services
  - Security annotations - Roles usage in Access rights management
- Development methodology
- Lessons learned

What is EHR?
Electronic Health Record (EHR)

- Defined as a collection of an individual patient's health records in digital format.
- These records contain patient medical information from multiple sources, accessible from any location by any provider caring for the patient.
- The information is continuously updated and always current.

Definition of EMR

Electronic medical record (EMR)

- is a patient medical record that contains documents in an electronic form and functions including:
  - Patient demographics
  - Medicine and allergy lists, and immunization status.
  - Scheduling, retrieval and archiving of laboratory and other tests
  - Appointment scheduling, patient reminders of follow up appointments, test completion, preventive health practices…
  - Claims and payment processing

Characteristics of EMR

- EMR integrates these elements so that patient data is available to each component, for example, billing modules, appointment reminders and test ordering systems
- EMR software allows viewing of data by any other **compatible** system
- EMR is commonly generated and maintained by one care provider (physician) or institution (clinic or hospital)
- a patient may have **several** EMRs

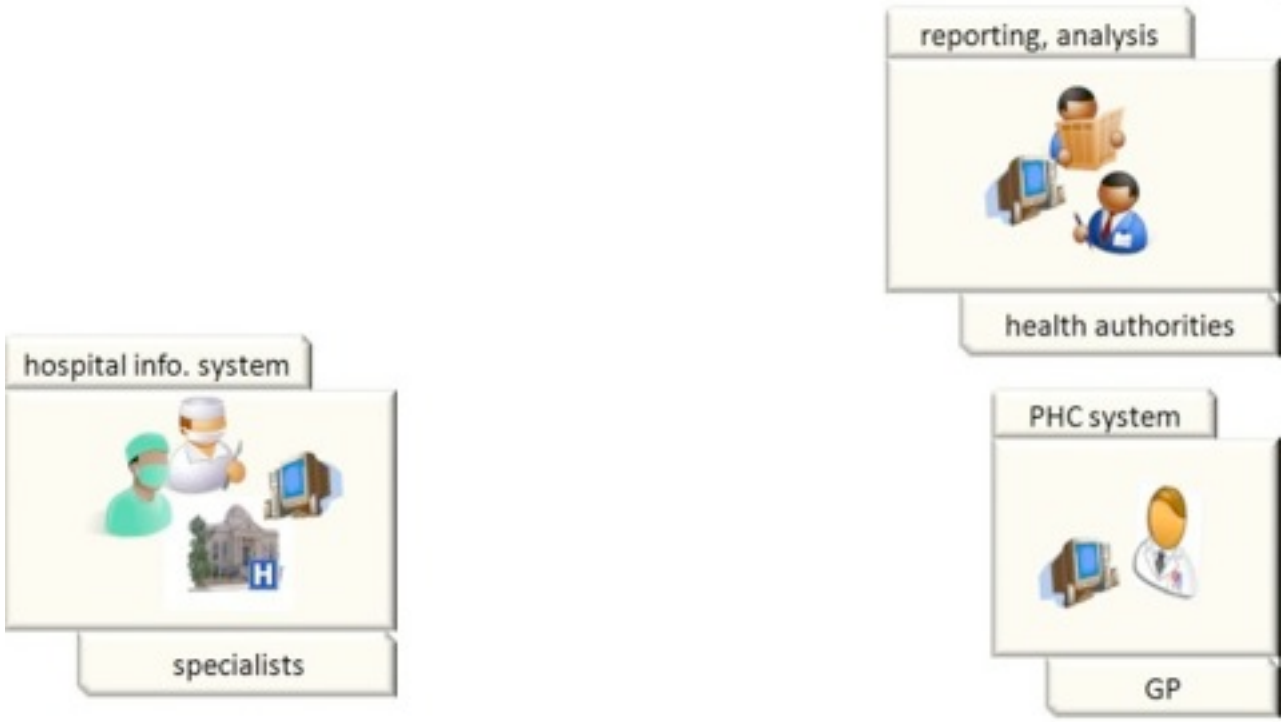
The difference between EMR and EHR

- The EHR is different from the EMR, as it is **not limited to a single healthcare enterprise** (*Waegemann, C. Peter, MRI, 2002.*)
- EHR is **accessible from any location** by any provider caring for the patient
- EHR is a set of patient medical information **from multiple sources**
- Secondary usage: EHR allows collection of data for uses other than for direct patient care: quality improvement, resource management, and public health communicable disease surveillance.

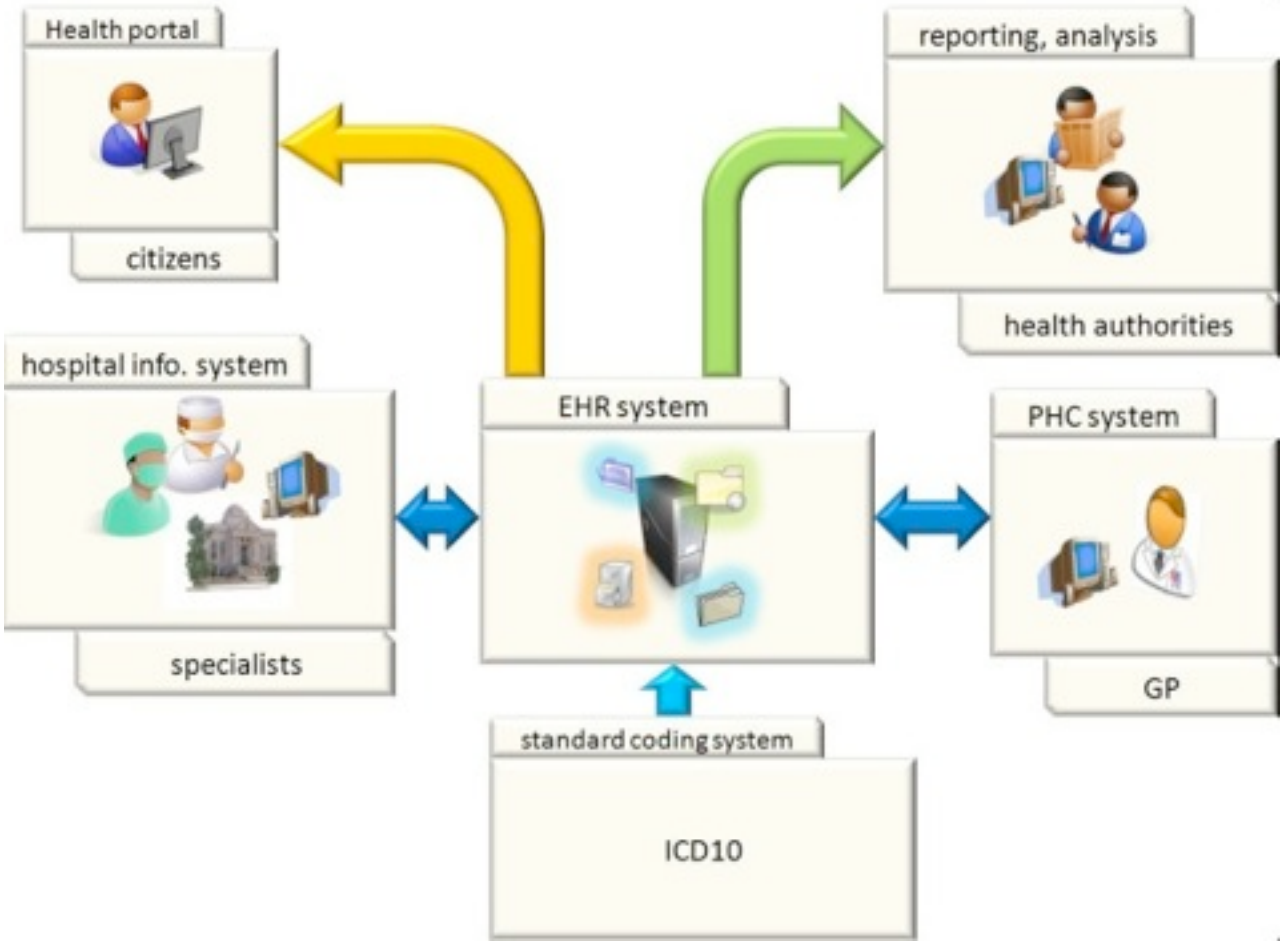
Characteristics of EHR

- Main features:
  - a provider-based view of that patient's health history
  - a method for clinical communication and care planning among the individual healthcare practitioners serving the patient
  - serves as the legal document describing the healthcare services provided
  - a source of data for clinical, health services, outcomes research, and public health

National level, before EHR



National level, with the EHR



The EHR project
Project information

- Started in 2005, ended in August 2008
- Financed by the EU, through the European Agency for Reconstruction and Euro Health Group
- Project ID: EuropeAid/117681/D/SV/YU
- Aid recipient country: Serbia

EHR System features

- EHR is a double-faced system by its very nature:
  - patients request absolute privacy protection of their data, but at the same time
  - medical professionals need fast and easy access to the particular record
  - As well, statistical reporting has a need for the actual data, but it has to be done in such a way that it would not put privacy at risk
  - The whole system, EHR plus the machine consumers, is distributed and heterogeneous- multiple subsystems, technologies, data sets

Standards applied

- Nation – wide EHR has to be based on standards, including: common coding systems, standard data sets and communication protocols, uniform data representation, as well as rules for protection of privacy and data access.
- Project charter only demanded usage of **Health Information Systems Architecture (HISA)** standard (**CEN 12967**)
- On top of this, interoperability with the International systems (like WHO) mandates usage of internationally adopted standards.

Standards applied (2)

Some additional international standards applied are

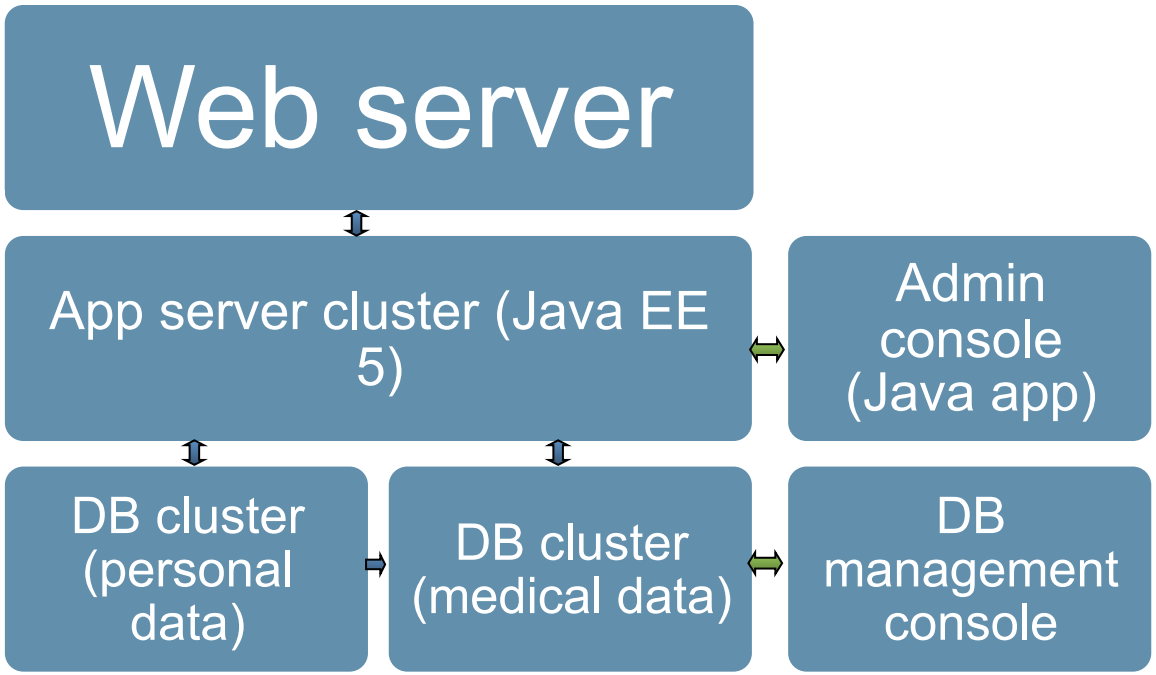
- International Classification of Diseases (ICD) 10th Revision (**ICD-10**) by the World Health Organization,
- The International Classification of Primary Care (ICPC) version 2 (**ICPC-2**) by WONCA International Classification Committee (WICC)
- privacy sensitivity levels of **CEN TC251 13606** EHR standard
- system provisions for later adoption of **SNOMED CT** and **GALEN** coding systems

The system is also multilingual

Usage scenarios

- All local systems were reused by enabling them to communicate with the EHR system
- These systems became secured, authenticated and authorized machine producers (of new medical data) and consumers (of existing medical data) of the EHR system
- The average user continued working on the particular system he or she is familiar with
- The novelty is in the ability to view full medical data about the patient, not just the one generated by their medical practice, or hospital

Basic Architecture of EHR core



Mapping of Java EE 5 features to EHR-S mechanisms

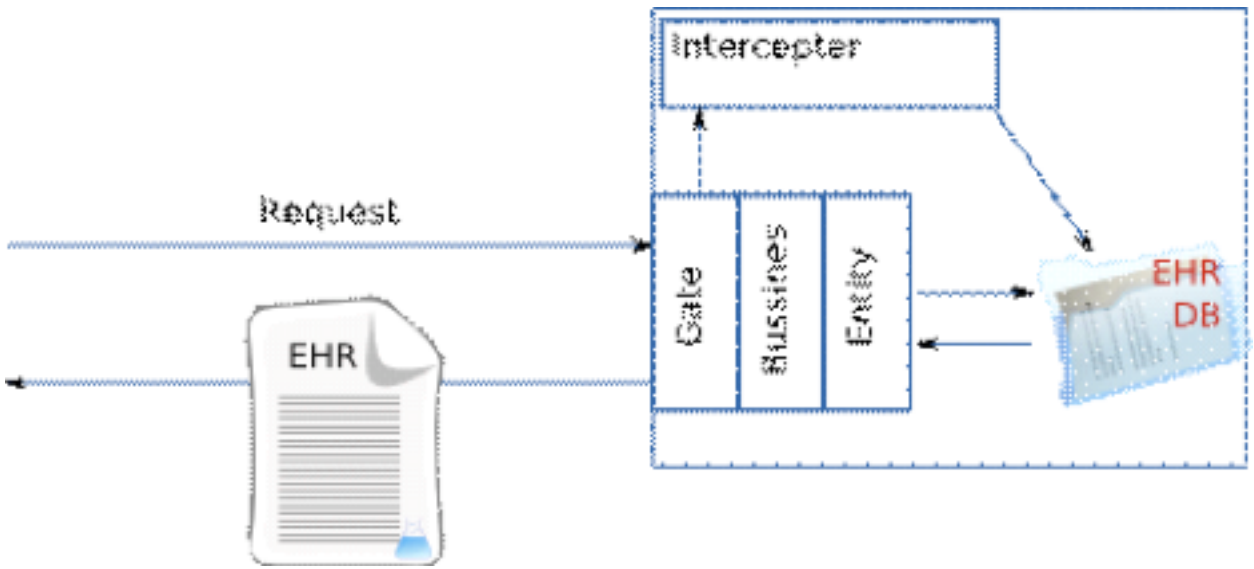
- We have used mapping of specific Java EE 5 features in solving particular process problems:
  - interceptors in role management
  - dependency injection in rules management
  - massive use of metadata annotations in persistence framework for database maintenance and schema management
  - web service for interoperability and communication management
  - annotation for security purposes

Interceptors

- useful for implementing role management and different business rules. Examples:
  - checking if some Health Care Professional is related to a Health Care Facility
  - ensuring that a prescribed medication is listed as a registered one
  - make sure that the patient is not historically allergic to prescribed medication
- those rules could be implemented in the business logic layer of the EHR-S model, but then it would be hard to change and maintain

Interceptors

The general idea of Interceptors usage in EHR



Dependency injection

- Benefit: add new features in the middle of EHR-S model with only a few changes
- Example: we want to access instances of classes **SubjectOfCare**, **HealthCareOrganisation** and **HealthCareProfessional**, just by adding @EJB dependence injection and use it when needed

```
public class GateBean implements GateLocal, GateWebService{
@EJB
 SubjectOfCareFacadeLocal socBean;
@EJB
 HealthCareOrganisationFacadeLocal healthCareOrganisationFacade;
@EJB
 HealthCareProfessionalFacadeLocal healthCareProfessionalFacade;
```

Persistence framework

- EHR-S implementation is using a relational database as the permanent data store.
- The persistence framework manage mapping between the database and the objects.
- The core EHR system has more than 50 tables in the data base (without the coding subsystem).
- We used annotations on Entity classes.
- The result is at least 30% coding reduction for the core EHR system.

Persistence framework
Example 1/2

```
@Entity
@Table(name = "ADDRESS")
@SequenceGenerator(name = "AddressSeq", sequenceName = "ADDRESS_SEQ", allocationSize = 1)
public class Address implements Serializable {
        @Id
        @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "AddressSeq")
@Column(name = "ID", nullable = false)
        private String id;
        @Column(name = "STREET", nullable = false)
        private String street;
@Column(name = "STREETNUMBER", nullable = false)
        private String streetNumber;
//…nxt slide
```

Persistence framework
Example 2/2

```
//…prev slide
@JoinColumn(name = "CDCOUNTRY_CODE", referencedColumnName = "CODE")
        @ManyToOne
        private CDCountry country;
        @JoinColumn(name = "PERSON_ID", referencedColumnName = "ID")
        @ManyToOne
        private Person person
```

Web services

- We used Web Services Description Language (WSDL)
- Integrated with few major local Health ICT vendors – their systems are based upon
  - J2EE (“ICT Sphera”, formerly “Omnilogika”)
  - .NET (EHRViewer, thank you, project “Tango” !)
  - FoxBase/Clipper (Nis Medical Centre)
  - Delphi (“Zip Soft”)
  - Progress (“Microsys”)

Web services

Example: Class with the functionality:

- @Stateless
- @WebService(endpointInterface = "ehr.gate.session.GateHealthWebService")
- @Interceptors({ HealthAuditInterceptor.class })
- public class GateHealthBean implements GateHealth, GateHealthLocal, GateHealthWebService {
- …

Web services

Example: Interface for WS, definitions of available

```
@WebService
public interface GateHealthWebService {
        @WebMethod(action = "getHR")
        HRDoc getHR(Long hcpId, Long hcfId, String socJmbg);
        @WebMethod(action = "commit")
        public String commit(Long hcpId, Long hcfId, Long socId, Medical medicalInformation);
}
```

Security annotations

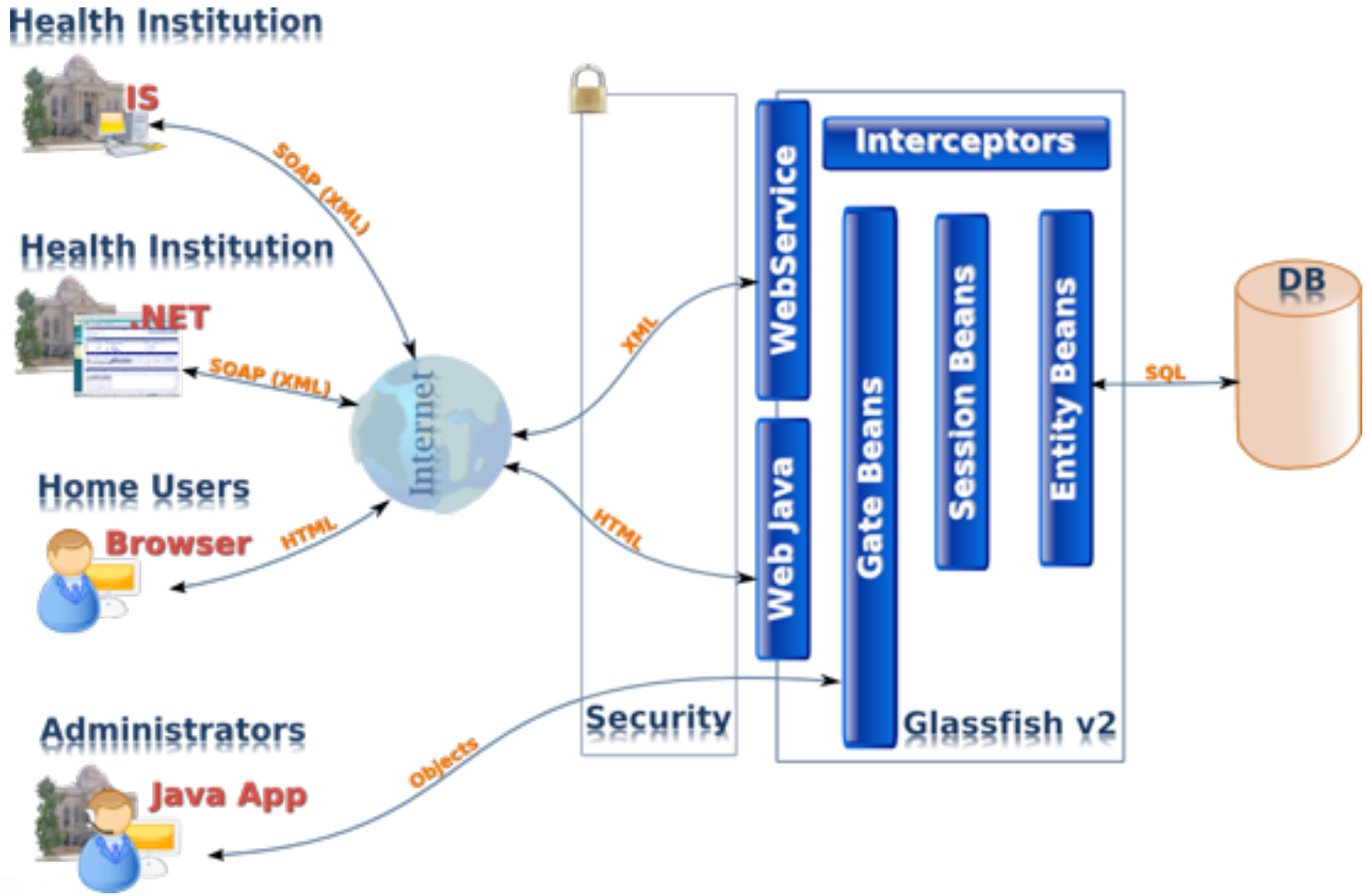
EHR general privacy and security issues

- Medical data are private and **highly confidential**
- Any breach or loss of privacy would undermine public confidence into the system
- Security of EHR must meet or exceed the standard that should be applied to paper records.
- Information security policy determines who may access what information
- Depends on the legal system and cultural values
- Therefore, one solution does **not** fit all; there are **many possible solutions**

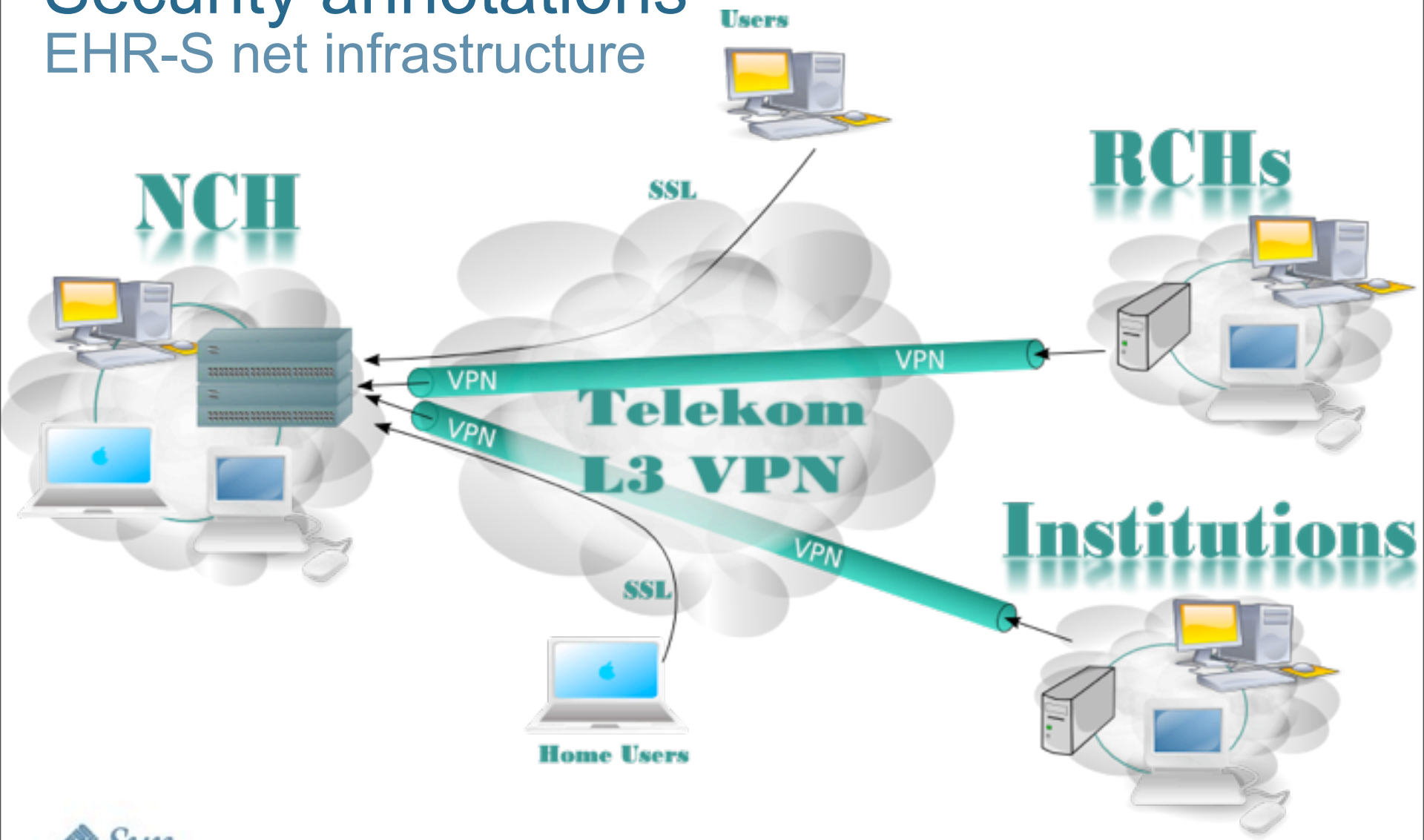
Security annotations
EHR-S authorization example

- users of the EHR are authorized to have access and perform actions on "**need to know**" basis
- Authorization of EHR-S users is dynamic: this means that what the authenticated user can do (is authorized to do) depends on:
  - who the user is (his ID and role)
  - what is the object of his action (which particular EHR is the object of his action), and the
  - the **relationship** of **the user** with **the object** of his action at the **particular** given **moment** of the action request

Security annotations
EHR-S general measures



Security annotations
EHR-S net infrastructure



Security annotations
Access rights

- EHR system is containing **highly confidential** data, utmost attention to the access rights is imperative.
- At the other hand, it is hard to manage access rights simultaneously with the functionality development.
- With the EJB3 we could add annotations to the methods and define access right independently from functionality development.

Security annotations (2)

Example: Declaring a role that could be used in a

- @Stateless
- @DeclareRoles({“administrators”,”hcp”,”emergency”, “nurse”, “selectedHcp”})
- @WebService(endpointInterface = "ehr.gate.session.GateHealthWebService")
- @Interceptors({ HealthAuditInterceptor.class })
- public class GateHealthBean implements GateHealth, GateHealthLocal, GateHealthWebService {
- ...

Security annotations (3)
Example: Simple role protection of a method

- @RolesAllowed({“hcp”})
- public void addHealthIssue() {
- ...
- }

Development methodology
Decisions

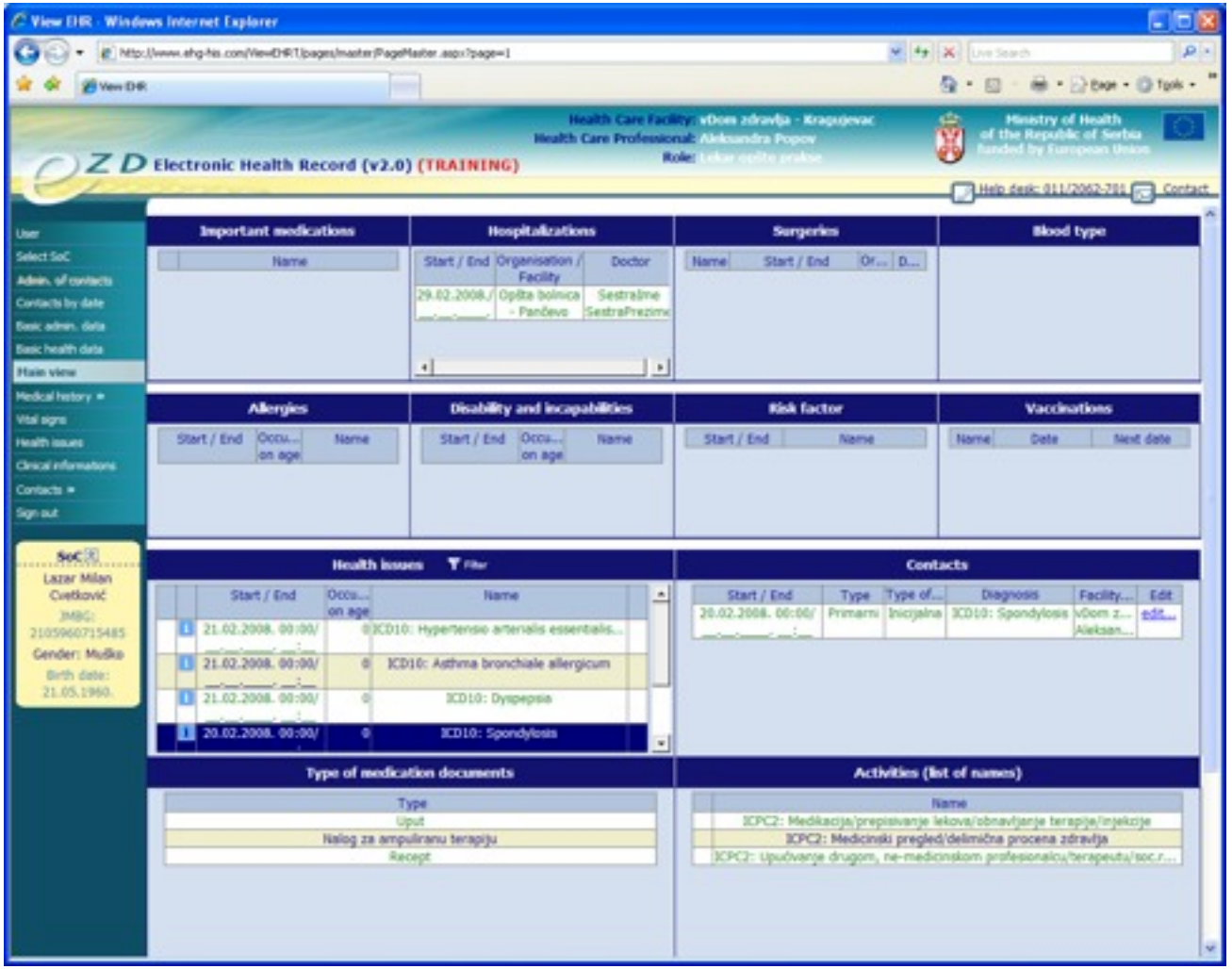
- Risk assessment directed a way to mitigate threats, and major points were Agile development methodology and Java EE 5 platform.
  - Decision to use Java EE 5 was a risky one - relative novelty, however a payoff was double: more modern platform with longer anticipated usability life, and way less programming in sense of KLoC.
  - Agile methodology enabled rapid progress, so within a few months a pilot implementation was installed.

Development methodology
Quality Assurance

- We have reused standalone .NET – JavaScript demo EHR viewer and converted it into prototype machine consumer.
  - refactored into the Web Services machine client for the EHR system.
  - This way we could always test how WS works and the users could also test the functionality, giving us a very quick testing and feedback loop.
  - we documented how to integrate with .NET and J2EE, and made it available to all health - domain related software vendors. This in turn made possible integration with the wide range of systems.

Development methodology

Quality Assurance – EHR Viewer – sanitized data

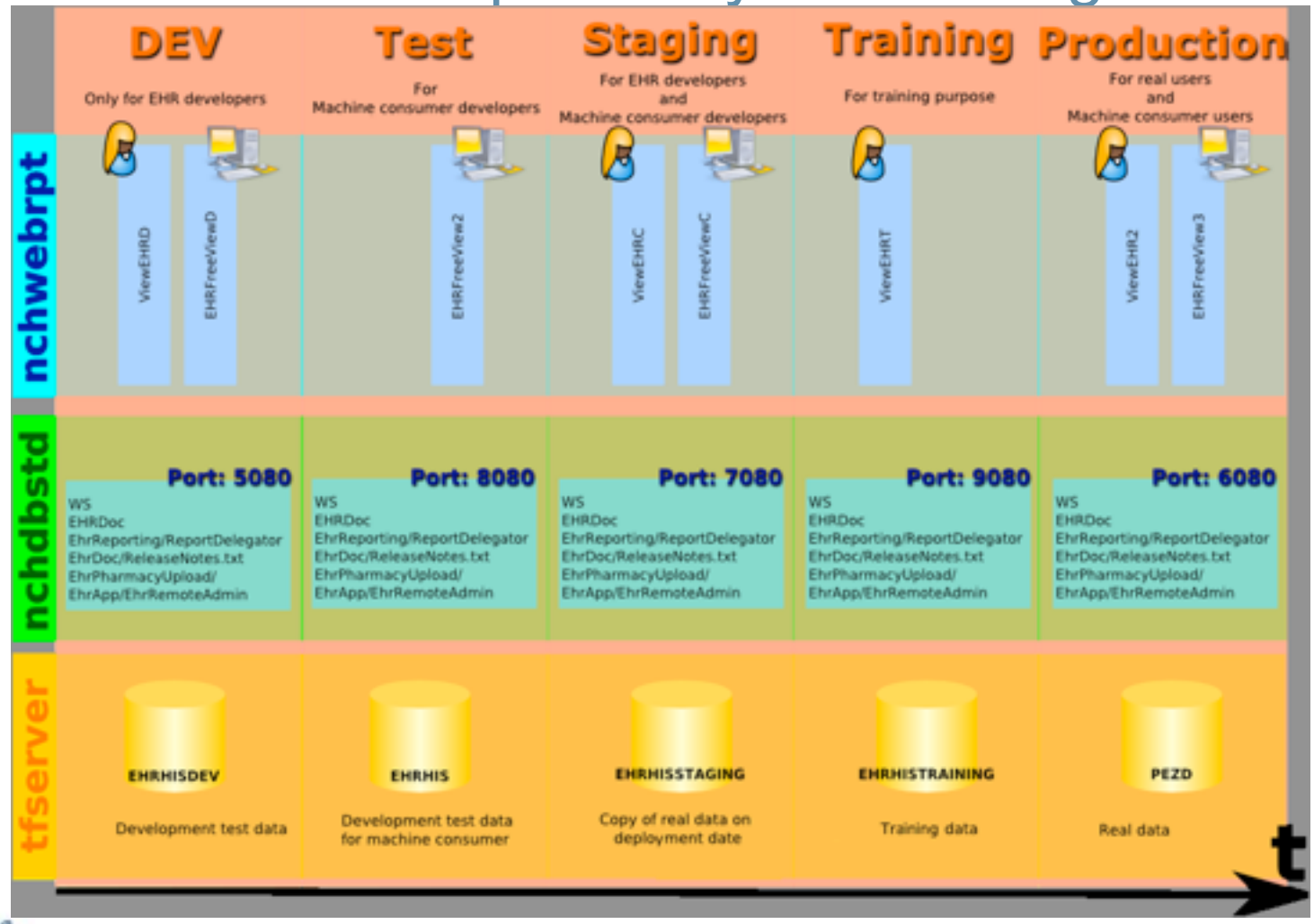


Development methodology
Integration and interoperability risk management

- gradual introduction of changes and the constant quality assurance. Five separate environments created, and all the changes had to pass through in order to get into the production. Stages are:
  - “dev” for development, and internal “core” developers only; after passing “dev” phase, changes get into the
  - “test” environment where developers of the local ICT solutions have the ability to test the interoperability;
  - “staging” represent the phase where any gap is analyzed and resolved;
  - “training” phase when end users get the training;
  - the “production” stage, the system is operational.

Development methodology

Integration and interoperability risk management



Lessons learned

- Java EE 5 features maps exceptionally well to system mechanisms needed in large – scale systems such as Nation-wide EHR System
- Clear development and testing process is crucial for the quality and timeliness of the system development
- Thanks to Java EE 5 platform, large scale EHR-S development is less the technical problem; risks other than technical (governance, processes, project management) are becoming more significant

Thank You

S.B.Stakic
sstakic@AdvancedSystemsGuild.com
D.Ljubojevic
dlj@AdvancedSystemsGuild.com

www.AdvancedSystemsGuild.com