Implementing a Knowledge-driven Hierarchical Context Model in a Medical Laboratory Information System

Jaan Pruulmann and Jan Willemsen
Tartu, Estonia

ICGCI 2008
July 27, 2008 - Athens, Greece
Contents

- Background: where the system was built
- Problem statement
- Solution ideas
- Building of model core
- Examples of how it functions
- Some preliminary results
Background objectives

- In result of our
  - Soviet past
  - Very intensive evolution
  - Today's open society

- We have big variety in
  - Doctors' background
  - Laboratory equipment
  - Laboratory processes
  - Computer systems

Need to cope with coexistence of old and new understandings, equipment, technology etc.

Small country – the amount of everything is tractable
## Background (some numbers)

<table>
<thead>
<tr>
<th>Clients</th>
<th>470 medical organizations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 different types of software, including 4 with capability for some electronic communication</td>
</tr>
<tr>
<td></td>
<td><strong>1250 doctors</strong> (clinicians, surgeons, anaesthetists, family doctors etc.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technology</th>
<th>About <strong>800</strong> individually described tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>55 different, partially overlapped, <strong>code sets</strong></td>
</tr>
<tr>
<td></td>
<td>54 different, partially overlapped <strong>technologies</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Structure</th>
<th><strong>Three 24x7 laboratories:</strong> one universal (in maternity ward), two specialised (clinical chemistry and haematology)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Nine business-hours laboratories</strong> are specialized to specific testing technology (e.g. immunology) or to specific profile of local hospital (e.g. oncology)</td>
</tr>
<tr>
<td></td>
<td>All have evolved in own <strong>local society and context</strong></td>
</tr>
</tbody>
</table>
Requirements for LIS

- LIS has to serve the common laboratory functioning 24x7 without breaks.
- LIS has to support
  - Structural reorganization of laboratory itself
  - Renewing testing technologies and equipment
  - Evolution of client computer systems and software
  - Cooperation with external laboratories while being continuously in use
A dream about solution

- LIS should be an **intelligent system**, being capable to adapt oneself to expected changes.
- It has to have **detailed, dynamic knowledge** about:
  - Nature of laboratory processes
  - Individual laboratory technologies and equipment
  - Individual laboratory workers (co-workers)
  - Individual clients and their needs to laboratory
  - External systems it communicate with
What is a medical laboratory?

- **Laboratory**
- Tests
- Order + specimen
- Result
- Doctor (or a HIS)
Hierarchical model of laboratory

Doctor (or a HIS)
Hierarchical model of laboratory

K8

K6

K3

K2

K4

Bin

Doctor (or a HIS)
Hierarchical model of laboratory

Doctor (or a HIS)
Hierarchical model of laboratory

Another laboratory

A part of laboratory

Doctor (or a HIS)
Hierarchical model of laboratory

Another laboratory

A part of laboratory

Doctor (or a HIS)
Structure of LIS
Structure of LIS

Human user

Remote HTTP(s)

HIS, LIS, Analyser, LabEquipment etc.
In real world, the LIS communicates with
Organization  Specimen  Client
Doctor  Agreement
Technician  Order  Container

HIS  Equipment  PC  LIS

It sees directly
The real solution concept

- LIS has its own conceptual model about real world around it including sub-models about
  - Laboratory medicine
  - Laboratory technologies
  - Laboratory organisation, including LIS users
  - Clients and their organisation and users

- LIS acts as an actor who simultaneously communicates with other actors
  - Human users
  - External systems
The model bases on hierarchies

**Taxonomical hierarchy**
- A natural way to describe *abstract concepts*. A property of a taxon is effective for all its sub-taxa

**Organisational hierarchy**
- **Rights** are partially delegated (or dispersed) from a level to some taxon on next lower level. **Responsibilities** corresponding to rights are delegated, too

**Compositional hierarchy**
- Describes how the *things* are composed from *parts*
Why hierarchies?

- They base on common tree abstraction that is simple in implementation
  - A tree *can be evolved* easily *'on the fly'* when a new detail is reported by an external system or user
- Unlimited detailing is possible
- They are simple to understand, too

But the world is *not so simple*. Is it?
A new abstraction can be created only on basis of some taxonomical difference according to abstractions existing before.

So all our abstractions can be in a common tree.

A taxon has its own id, code, description and parent.
Class model of context engine

- A subtree of taxa constitutes a whole taxonomy corresponding to its root taxon
- E.g. any code set is a simple taxonomy
Class model of context engine

Abstractions (taxa) may be described by a number of valued attributes of particular types
An attribute type is an abstraction, too, and hence a taxon in the taxonomy of attribute types.

Attribute types may be hierarchically detailed.
Attribute types (example)
Coded attributes are directly linked with the taxon that corresponds to the attribute value.

The organisational and compositional hierarchies are described by links with corresponding types.
A taxon can have a number of different type names in a human language or language dialect.
Taxonomy languages (example)
Context algebra of codes

- Every taxon has its own code.
- Code is unique in the set of sibling taxa.
- FPC – Full Path Code – list of codes from root taxon to the taxon itself.
- We can split a FPC into two parts at any tree arc
  \[ FPC = FPC_{\text{context}} + \text{CODE}_{\text{in-context}} \]
  - Where \( \text{CODE}_{\text{in-context}} \) is code of the taxon when the taxonomy \( FPC_{\text{context}} \) is determined by context.
- A taxon has different codes in different contexts.
Context algebra of codes (example)


Full Path Code: it can be split and combined

(context) • (address in context)

addr-Estonia • Tartu-Tähe-4-B-2001

addr-Estonia-Tartu • Tähe-4-B-201

addr-Estonia-Tartu-Tähe-4 • B-201
Context algebra of codes (example)


Common location

(context) • (address in context)

addr-Estonia • Tartu-Tähe-4-B-2001

addr-Estonia-Tartu • Tähe-4-B-201

addr-Estonia-Tartu-Tähe-4 • B-201
Context algebra of codes (example)


Root – addr – Estonia – Tallinn - Akadeemia – 21

Common location (context) • Destination address (address in context)

addr-Estonia • Tartu-Tähe-4-B-2001

addr-Estonia-Tartu • Tähe-4-B-201

addr-Estonia-Tartu-Tähe-4 • B-201
Direct links
Context in work
Some preliminary results

<table>
<thead>
<tr>
<th>Category</th>
<th>Release-time</th>
<th>After 12 months</th>
<th>Increase %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxons</td>
<td>3377</td>
<td>6038</td>
<td>78.8</td>
</tr>
<tr>
<td>Non-text attributes</td>
<td>20010</td>
<td>28050</td>
<td>40.2</td>
</tr>
<tr>
<td>Text attributes</td>
<td>4928</td>
<td>7844</td>
<td>59.2</td>
</tr>
<tr>
<td>Links</td>
<td>29586</td>
<td>46866</td>
<td>58.4</td>
</tr>
</tbody>
</table>

1. Evolution of the taxonomic hierarchy

<table>
<thead>
<tr>
<th>Category</th>
<th>Release-time</th>
<th>After 12 months</th>
<th>Increase %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers</td>
<td>86</td>
<td>63</td>
<td>-26.7</td>
</tr>
<tr>
<td>Materials</td>
<td>231</td>
<td>317</td>
<td>37.2</td>
</tr>
<tr>
<td>Cont. × Mat.</td>
<td>19866</td>
<td>19971</td>
<td>0.5</td>
</tr>
<tr>
<td>C-M Links</td>
<td>755</td>
<td>312</td>
<td>-58.7</td>
</tr>
</tbody>
</table>

2. Self-optimization of the hierarchy
Results/Summary

To construct a knowledge driven IS with AI, the following have proven to be useful for it:

1. The **regular** knowledge mode cross-linked hierarchies and taxonomies.
   - The context is focused by data in communication
   - The abstraction level is dynamically adequate

2. The concept of **IS** where it is an actor who just communicates with other actors
   - No hard-coded descriptions of process but reacting to input events in inputs' context
   - Figures out the role of observer
Thank You.