Towards a Multi Meta-modelling Approach for Developing Distributed Healthcare Applications

Fazle Rabbi$^{1,2}$, Yngve Lamo$^1$, Lars M. Kristensen$^1$ and Ingrid C. Yu$^2$

$^1$ Department of Computing, Mathematics, and Physics
Bergen University of Applied Sciences

$^2$ Department of Informatics
University of Oslo
Healthcare Systems and IT

- **Involves the organisation of:**
  - **People:** healthcare personnel, patients, ...
  - **Institutions:** hospitals, emergency services, doctor’s office, ...
  - **Resources:** equipment, medical supplies, biological material, ...

- **IT systems for healthcare services must:**
  - **Integrate** with external healthcare systems and services.
  - Support electronic forms of capturing *patient information*.
  - **Adapt** to different settings and the use of different services.
  - Support workflows following *prescribed standards*.
  - Be **adaptable** to changes in rules and regulations.
Blood Transfusion Application

- Application being developed in collaboration with Helse Bergen and Helse Vest IKT:

  1. **Read** identity of patients from wristbands.
  2. **Access** Electronic Health Record (EHR) to determine blood type.
  3. **Print** tags for blood samples to be sent to the laboratory.
  4. **Order** blood from blood bank based on patient identity or tags.
  5. **Send** screening results from laboratory to the blood bank.
  6. **Match** patient identity and blood samples (safety).
  7. **Record** any complications related to blood transfusion.

- A **distributed application** required to support and conform to a highly regulated work process.
Model Driven Engineering

- A model-centric approach to software engineering:

  **Meta-modelling:**
  Defining domain-specific modelling languages.

  **M2M/M2T transformations:**
  Model refinement, model execution, code generation.

  **Simulation and verification:**
  Validate the correctness of models and designs.
Distributed Healthcare Apps

- Modelling a distributed healthcare application requires multiple modelling languages:
  - **Workflow modelling** – for modelling the underlying work process of the application.
  - **Data entity modelling** – for modelling the data and information being processed.
  - **Message passing modelling** – for modelling the exchange of data between processes.

- **Coordination** for coherent linkage of models belonging to the different modelling languages.

Diagram Predicate Framework

- Formal foundation for meta-modelling, model coordination, and model transformation:
  - A graph-based diagrammatic approach to meta-modelling.
  - Conformance (typing) based on graph homomorphisms and on predicates expressing additional model constraints.

- Supported by the DPF Workbench and WebDPF:

  Eclipse EMP-based
  [ [http://dpf.hib.no ]

  Web-based
  [ [http://data2.hib.no:8080/dpf2.0/index.htm ]

[NIK’2015 - 7]
Overview: Multi Meta-modelling

Program in dynamic logic

Coordination

Workflow metamodel

Workflow model

Entity model 1

Entity model 2

DataSrc

Represented by
Workflow Modelling

- DERF language* extended with system locations, message passing, and condition predicates:

Workflow Modelling

- DERF language extended with **system locations, message passing, and condition predicates**:

  Workflow routing
Workflow Execution

- Execution of the extended DERF workflow models is based on model transformations:

<table>
<thead>
<tr>
<th>p</th>
<th>Visualization</th>
<th>Coupled transformation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>[sequence]</td>
<td>X →₇ Y [seq]</td>
<td>X →₇ Y [seq]</td>
</tr>
</tbody>
</table>

DIAGRAM:

- Scan Patients Wristband @App
- Get Patient Info @EHR
- Check blood Type @App

- [seq] connection between steps
Entity Modelling

- Subsystems may have different entity models - but rely on a common entity meta-model:
Message Passing

- Based on **typed graphs** matched against subsystem entity instance models:
Process Descriptions

- **Dynamic logic programs** are used to specify the detailed behaviour of processes.

- **Example** - Checking blood type:

\[
\alpha := \alpha_1; \neg (z = \text{""})? \alpha_2; (z = \text{""})?
\]

\[
\alpha_1 := \text{Patient}(p); \text{patientID}(\text{pid}); \text{src}(\text{pid}, p); \text{trg}(\text{pid}, y)
\]

\[
\alpha_2 := \text{String}(z); \text{bloodType}(bt); \text{src}(bt, p); \text{trg}(bt, z)
\]

\[
\begin{align*}
y &= 101 \\
p &= \text{"Adam"} \\
z &= \text{"AB+"} \\
\text{pid} &= p1 \\
\text{bt} &= b1
\end{align*}
\]
Conclusions and Future Work

- A multi-modelling approach for the development of distributed healthcare applications:
  - Workflow modelling and workflow execution.
  - Entity modelling and process descriptions.
  - Message passing and process locations.
- Evaluation on a blood transfusion application.
- Ongoing and future work:
  - Modelling language extensions: exception handling, timing constraints, modules and containments (abstraction).
  - Simulation-based validation of the application design: process conformance and user experience.
  - Formal verification: workflows and process descriptions.
  - Code generation: interfaces for the application subsystems.