Modeling Data for Business Processes

Yutian Sun¹  Jianwen Su¹  Budan Wu²  Jian Yang³

¹ University of California, Santa Barbara
² Beijing University of Posts and Telecommunication
³ Macquarie University
Repetitive Coding due to Process Change
e.g.: KF (kingfore.net) Heating Equipment Repairing Co.

A repair business process*

- Repair Application
- Application Review
- Repair Assignment
- Doc. Archive
- On-site Repair
- Post-repair Visit
- Database

Change is very difficult

Hard-coded SQL

A deletion ⇒ rewriting of 4% of code ⇒ all for DB access
20% of code are for DB access

*Developed using jBPM engine. Expressed as a BPMN process [OMG standard]
Business Entities w/ Lifecycle are not a Solution

A repair *business entity with lifecycle* [Nigam-Caswell 03]

The problem exists: *what is the relationship?*

```
<table>
<thead>
<tr>
<th>Repair</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>RID</td>
<td>CName</td>
</tr>
<tr>
<td></td>
<td>Addr.</td>
</tr>
<tr>
<td>Services</td>
<td></td>
</tr>
<tr>
<td>SID</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Repairperson</td>
</tr>
</tbody>
</table>

Database

- Repair Application
- Repair Assignment
- On-site Repair
- Doc. Archive
- Post-repair Visit

Application Review

X
A Main Contribution

Business entities with lifecycles

- With the data mapping
  - Code can be automatically generated → help changes
  - Help data analysis

This paper formulates a language for mapping business entities and database
Outline

• A Motivating Example
• Entity-Data Mapping Language
• Equivalence to a subset of Clio
• Updatability
• Isolation
• Conclusion and Future Work
Business Entities with Lifecycle

Repair

Customer

Services

Parts

MID

PName

ID

RID

CName

Addr

SID

Date

Repairperson

Set

Keys

Primitive Attributes

Business entity

Lifecycle

Can be Petri-Nets, FSM, rule-based style, etc.

Repair Application

Repair Assignment

On-site Repair

Post-repair Visit

Application Review

Doc. Archive

X

X

X

X

April 1, 2014
**Entity-Data (ED) Mapping Rules – IDs**

**RID:** RepairInfo.RIID

- Exists such a value in the DB

**RepairInfo**

- RIID | CustName
- R101 | David
- R102 | Peter

**User**

- UserName | Address
- Peter | 45 Main St
- David | 123 5th Av
- James | 78 Park St

**ServiceInfo**

- SIID | Date | RepairID
- S01 | 11/15 | R101
- S02 | 11/29 | R102
- S03 | 12/17 | R101

**Business entity**

**Database**
ED Mapping Rules – Primitive Attributes

Addr = Addr.Customer.RID
@RepairInfo(RIID).CustName
@User(UserName).Address
ED Mapping Rules – Keys

SID:
ServiceInfo.SIID
when
SID.Services.RID = ServiceInfo.RepairID
Clio Mapping and Equivalence

∀ rid, cn, add, sid, d,
RepairInfo(rid, cn), User(cn, add), ServiceInfo(sid, rid, d)
→ ∃ C, S, Repair(rid, C, S), C(cn, add), S(sid, d)

• Theorem: The ED mapping language is equivalent to a subset of the Clio mapping language

Entity-data mapping

Clio mapping

Conditions are syntactically defined
Outline

• A Motivating Example
• Entity-Data Mapping Language
• Equivalence to a subset of Clio
• Updatability
• Isolation
• Conclusion and Future Work
Updatability

• Can DB capture the updates of business entity instances, and vice versa?

\[ \mu : \text{mapping} \quad \delta : \text{updates} \]
Adding the new tuple for “Services” in business entities can be reflected as adding a new tuple for “ServiceInfo” in DB.
Not Updatable

When “Addr” is changed, there is no corresponding updates for DB to satisfy the mapping

RepairInfo

<table>
<thead>
<tr>
<th>RIID</th>
<th>CustName</th>
</tr>
</thead>
<tbody>
<tr>
<td>R101</td>
<td>David</td>
</tr>
<tr>
<td>R102</td>
<td>Peter</td>
</tr>
</tbody>
</table>

User

<table>
<thead>
<tr>
<th>UserName</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>45 Main St</td>
</tr>
<tr>
<td>David</td>
<td>123 5th Av</td>
</tr>
<tr>
<td>James</td>
<td>78 Park St</td>
</tr>
</tbody>
</table>

ServiceInfo

<table>
<thead>
<tr>
<th>SIID</th>
<th>Date</th>
<th>RepairID</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01</td>
<td>11/15</td>
<td>R101</td>
</tr>
<tr>
<td>S02</td>
<td>11/29</td>
<td>R102</td>
</tr>
<tr>
<td>S03</td>
<td>12/17</td>
<td>R101</td>
</tr>
</tbody>
</table>
Determining Updatability

- Syntactic conditions are provided to determine if a given set of ED mapping rules is updatable

- View updates
  - Often has no solutions
  - Focus on relational models
Outline

• A Motivating Example
• Entity-Data Mapping Language
• Equivalence to a subset of Clio
• Updatability
• Isolation
• Conclusion and Future Work
Isolation

• Will an business entity update affect other business entity instances?
Not Isolated

- Desirable: if “Addr” is billing address
- Undesirable: if “Addr” is repair address

Process Instance 1

Process Instance 2
Update Constraints

- Use update constraints to represent lifecycles:

```
CName = ⊥ ⇔ (Update, Services)
1 = 1 ⇔ (Update, Addr)
```
Determining Isolation

- With update constraints:
  - Upper bound: Exponential time w.r.t. # of attributes in the given business entities
  - Lower bound: have not studied yet

- No update constraints:
  - PTIME
Outline

- A Motivating Example
- Entity-Data Mapping Language
- Equivalence to a subset of Clio
- Updatability
- Isolation
- Conclusion and Future Work
Conclusion

• Repetitive coding effort
  • We define a mapping language
  • To generate code automatically

• The first approach to bridge the business process data and the database data

• Help the changes in business processes
Future Work

• A tool (Under development)
• Transaction issues
• Update models
• Propagate integrity constraints from database to business entities, and vice versa
• Other properties in addition to updatability/isolation
• Data mapping as a mathematical notion
• Process views
Thanks

Q&A