Synthesizing Information Systems: the APIS Project

Marc Frappier, Benoît Fraikin, Frédéric Gervais, Régine Laleau, Mario Richard

GRIL

Université Paris-12
Information Systems (IS)

- Typical example: ERP
- COTS ERP global market US$ 16.7 billion in 2005
- Canadian application development and systems integration CAD$ 4.464 billion in 2006
- Canada: 75,000 jobs in IS development
COTS

• Simplicity
  – buy knowledge, reduce development risk, (reduce cost)

• Rigidity
  – business process must conform to the COTS
  – little room for adaptation (hard to integrate new COTS versions)
IS Development : State of the Art

• Specification & Design
  – UML, mostly

• Implementation
  – J2EE, Hibernate, Spring, JDBC, SQL, ...
  – Web : Struts, JSF, AJAX, HTML, java scripts, ...
  – SOA, WSDL, BPEL, WSCI
  – MDA
IS development is expensive

- Technology is constantly changing
- Technology is overloaded with minute details to deal with
- Technology requires a long learning curve
Service systems

• Create value
• Precondition
  – cheap development costs!
• This is where we can contribute
The APIS project

• Automatic Production of Information Systems
• Objectives
  – Automatically derive information systems (IS) implementation from formal specifications
  – Validate IS specifications
  – Reuse IS specifications
  – Maintain IS specifications
  – Develop notations and tools to support these activities
Software Synthesis

• Automatically derive a software from a specification
Goal : Reducing Software Development Costs

Traditional Development

- acceptance testing
- requirements analysis
- design
- unit & integration testing
- coding

Synthesis

- acceptance testing
- requirements analysis

Goal
50 % effort reduction
Goal: Improving System Reliability

<table>
<thead>
<tr>
<th>Traditional Development</th>
<th>Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• test</td>
<td>• specification verification / validation</td>
</tr>
<tr>
<td>• not exhaustive</td>
<td>• code generation</td>
</tr>
<tr>
<td>• only shows the presence of errors</td>
<td>• fast interpretation</td>
</tr>
<tr>
<td>• does not prove the absence of errors</td>
<td>• proof of synthesis algorithms</td>
</tr>
</tbody>
</table>
Goal: Improving Adaptability

- Define specification patterns
- Allow them to be instantiated and personalised
- Reuse patterns from business to business
- How to manage integration between different versions?
The EB$^3$ Method

- Designed for *information system* specification
- hybrid notation:
  - process algebra
  - one state variable (system trace)
  - recursive functions for entity attributes
  - input-output rules
EB³ Support : APIS platform

Software Engineer

IS User

Input / output

Generate

Execute
The EB³ Method - Steps

1. Define business model (UML ER diagram)
2. Identify external events for entities and associations
3. Specify entity and association behavior using a process expression
4. Define entity attributes computation using recursive functions on traces
5. Define output using rules on traces and attributes
1 - Business Model

entity type

association

attribute

book

Acquire
Discard
ListBook

bookId
title

loan

Lend
Renew
Return
date

member

Join
Leave

borrower

memberId
name
nbLoans
maxNbLoans

action

1

*
2 – Actions (events)

Acquire( book : BookId, title : STRING ) : void

Lend( MemberId : MemberId, book : BookId ) : void

ListBookId( book : BookId ) : ( title : STRING, loanCnt : Nat)
3 - Entity Type Behavior: Book

\[
\text{book}(b : \text{BookId}) = \\
\text{Acquire}(b, _) \cdot \\
( \\
( \\
\text{loan}(_ , b)^* \\
\text{ListBookId}(b)^* \\
) \cdot \\
\text{Discard}(b)
\]
3 - Entity Type Behavior : Member

```
member(m : MemberId ) =

Join(m, __, __) •
( ||| b : BookId : loan(m, b)* ) • Leave(m)
```
3 - Association Behavior : Loan

\[
\text{loan}(\text{bId}: \text{BookID}, \text{mId}: \text{MemberID}) = \\
\text{nbLoans}(\text{mId}) < \text{maxNbLoans}(\text{mId}) \rightarrow \\
\text{Lend}(\text{mId}, \text{bId}) \cdot \\
( \text{nbLoans}(\text{mId}) < \text{maxNbLoans}(\text{mId}) \rightarrow \\
\text{Renew}(\text{bId}) )^* \cdot \\
\text{Return}(\text{bId})
\]
3- Main Process

main =
    ( ||| b : BookId : book(b)* )
||
    ( ||| m : MemberId : member(m)* )
4 - Entity Attribute Specification

\[
\text{nbLoans}(\text{mld} : \text{MemberId}) = \\
\quad \text{Join}(\text{mld}, _) : \text{return } 0, \\
\quad \text{Lend}(\text{mld}, _) : 1 + \text{nbLoans}(\text{mld}), \\
\quad \text{Return}(\text{bId}) : \text{if } \text{borrower}(\text{bId}) = \text{mId} \\
\quad \quad \quad \text{then } \text{nbLoans}(\text{mld}) - 1, \\
\quad \text{Leave}(\text{mld}, _) : \bot; \\
\]

\text{end}
5 - Input-Output Rules

• Use plain SQL queries
• Developing an ER query language based on a set-theoretic model
Expressing Queries

• Use set-theoretic representation of ER model
  – sets
  – functions
  – relations
\[ \{ x, f_1(x), f_2(f_3(y)) \mid x \in key_1 \land y \in key_2 \land f_4(x) = f_5(y) \} \]

E1[x].key1, x.f1, E2[y].f3.f2 WHERE x.f4 = y.f5
E1[x].key1, x.f1, E2[y].f3.f2  WHERE  x.f4 = y.f5

translates into

SELECT E1.key1, E1.f1, E4.f2
FROM    E1, E2, E3, E4
WHERE
    E2.key2 = E4.f3
AND
    NOT EXISTS (SELECT * FROM E3 AS t1E3
                 WHERE t1E3.key1 = E1.key1 and
                       t1E3.key2 is null)
AND
    NOT EXISTS (SELECT * FROM E3 AS t2E3
                 WHERE t2E3.key2 = E2.key2 and
                       t2E3.key2 is null)
User Interface Specification

PageTest

age : nat $\neq$ 0;
nom : string $\neq$ null;
ageEnfants : (nat $\neq$)*
= {null;null;null};
OK : Bouton $? = 0$;
ANNULER : Bouton $? = 0$;
CONFIRMER : Bouton $? = 0$;
UI Specification

page selectMember
    { memberId : int ?; displayLoans : button; }

    if (displayLoans)
    {
        listOfLoans(
            memberId = memberId,
            loans = Library.getLoans(memberId))
    }

page listOfLoans
    { memberId : int;
      loans : (bookId : int, title : string, return : button) *
    }

    if (loans.return)
    {
        (Library.return(memberId, loans.selected.bookId));
        listOfLoans(loans = Library.getLoans(memberId))
    }
Example : Mail Reader
EB³PAI – Process Algebra Interpreter

\[
\begin{align*}
\text{EB}^3_1 & \quad \sigma \in \Sigma_e \cup \{\lambda\} \\
\sigma & \quad \xrightarrow{\sigma} \quad \square \\
\text{EB}^3_3 & \quad E_1 \xrightarrow{\sigma} E'_1 \\
E_1 & \quad | \quad E_2 \xrightarrow{\sigma} E'_1 \\
\text{EB}^3_2 & \quad E_1 \xrightarrow{\sigma} E'_1 \land E'_1 \neq \square \\
E_1 \cdot E_2 & \quad \xrightarrow{\sigma} E'_1 \cdot E_2 \\
\text{EB}^3_4 & \quad E_1 \xrightarrow{\sigma} E'_1 \land E'_1 = \square \\
E_1 \cdot E_2 & \quad \xrightarrow{\sigma} E_2
\end{align*}
\]
Transition Computation by Proof

\[ a.(b \mid c).d \quad \xrightarrow{a} \quad (b \mid c).d \quad \xrightarrow{b} \quad d \quad \xrightarrow{d} \quad \square \]

\[
\begin{align*}
\text{EB}^3_1 & & \quad b \in \Sigma_e \cup \{\lambda\} \\
\text{EB}^3_3 & & \quad b \quad \xrightarrow{b} \quad \square \\
\text{EB}^3_4 & & \quad (b \mid c) \quad \xrightarrow{b} \quad \square \\
\text{EB}^3_4 & & \quad (b \mid c).d \quad \xrightarrow{b} \quad d
\end{align*}
\]
EB³PAI : Algorithmic Complexity

• $n$ : number of entities in an entity type
• $s$ : size of specification text

$O(s + \log(n))$

• Transaction execution time for library system

100 ms
Attribute Specification

dueDate\(s : T(\text{main}), bId : bK_{-}Set\) : DATE \(\triangleq\)
match last\(s\) with
\(\perp : \perp,\)
Lend\((bId, mId, \text{type})\) : if type = Permanent
\hspace{1em} then \text{CurrentDate} + 365
\hspace{1em} else \text{CurrentDate} + \text{loanDuration}(mId)
\hspace{1em} end ,
Return\((bId)\) : \(\perp,\)
- : dueDate(front\(s\), bId);
TRANSACTION Lend(bId : bK_Set, mId : mK_Set,
   typeofLoan : Loan_Type)
VAR TEMP1 : INT
   SELECT CurrentDate + loanDuration INTO #TEMP1
   FROM member
   WHERE memberKey = #mId;
UPDATE loan SET borrower = #mId
WHERE bookKey = #bId;
IF SQL%NotFoudn
THEN
   INSERT INTO loan(bookKey,borrower)
   VALUES (#bId,#mId);
END:
IF typeofLoan = "Permanent"
THEN
   UPDATE loan SET dueDate = CurrentDate + 365
   WHERE bookKey = #bId;
   IF SQL%NotFoudn
   THEN
      INSERT INTO loan(bookKey,dueDate)
      VALUES (#bId,CurrentDate + 365);
   END;
ELSE
   UPDATE loan SET dueDate = #TEMP1
   WHERE bookKey = #bId;
   IF SQL%NotFoudn
   THEN
      INSERT INTO loan(bookKey,dueDate)
      VALUES (#bId,#TEMP1);
   END;
END;
COMMIT;
Future Development

- ELTS: extended labeled transition systems
- Validation of $EB^3$ specification
- Reuse of $EB^3$ specification
ELTS

\[ E \overset{\triangle}{=} \| y : V : A(y, \_ ) \cdot B(y)^* \cdot C(y) \]
Reuse

- process expressions
- attribute definitions
- inheritance
- instantiation
- patterns
1:N association
1-N association

\[
e_1(k_{e_1} : K_{e_1}) \triangleq \mathcal{P}^{e_1}(k_{e_1}) \cdot (\mathcal{M}^{e_1}(k_{e_1})^* || || k_{e_2} : K_{e_2} : a(k_{e_1}, k_{e_2})^* ) \cdot \mathcal{C}^{e_1}(k_{e_1})
\]

\[
e_2(k_{e_2} : K_{e_2}) \triangleq \mathcal{P}^{e_2}(k_{e_2}) \cdot (\mathcal{M}^{e_2}(k_{e_2})^* || || a(_, k_{e_2})^* ) \cdot \mathcal{C}^{e_2}(k_{e_2})
\]
Functional Security

• specify security policies at the business requirements level
  – “A broker must obtain approval from his client and then from senior manager before bidding on products of type X”
  – “In case of emergency, the emergency personnel may have access to the patient’s record without approval of treating doctor. The patient and the treating doctor must be notified of this access.”
Functional Security in SOA
Conclusion

• raise level of abstraction
  – reduce design, coding and testing
• generate code / symbolic execution
• reduce IS development cost and schedule
• increase adaptability of IS
• increase reliability
• increase verifyability
Limits and Perspectives

• still need to write some code for domain specific operators and functions
• efficiency of symbolic execution / generated code
• usability of notation
• one-size-fits-all for GUI
• need more tools ($$$$$)
• integration with legacy systems & new technology
• integration with IS development process
Industrial partners

• CGI Group
  – setting up a chair in IS synthesis
  – NSERC CRD to submit in May

• ÆBIS
  – domain modeling
  – knowledge management

• National Bank of Canada
  – NSERC strategic grant on functional security