Designing Compliant Service-based Business Processes with ECT

Natallia Kokash
Joint work with Christian Krause, Erik de Vink and Behnaz Changizi
Model-driven development (MDD)
Application of Eclipse Coordination Tools (ECT) to process verification
Formalization of process models (control flow)
Formalization of compliance requirements
Modeling data flow
Model checking data-aware Reo models with mCRL2
Conclusions and Future Work
   Models for QoS analysis
COMPAS project

- **COMPAS**
  - Compliance-driven Models, Languages, and Architectures for Services

- **Goal**
  - Ensure dynamic and on-going compliance of software services to business regulations and user requirements

- **Methodology, terminology and research results**
  - Compliance is conformity in fulfilling compliance requirements
  - Compliance requirement is a constraint or assertion that results from the interpretation of the compliance sources
  - Compliance source is a document that is the origin of compliance requirements (e.g., SOX, HIPAA, licenses)
  - Compliance rule is an operative definition of a compliance requirement
  - Business process is a composition of activities into a structured order that implements the procedure to be followed in order to achieve a business goal
  - Behavioral model is a description of how an actor (e.g., stakeholder, service) acts or interacts with other actors

- **Check whether compliance rules hold for formal behavioral models of business processes**
Model-driven development

- A software development methodology which focuses on creating models (i.e., abstractions), close to domain concepts rather than computing (i.e., algorithmic) concepts
  - Maximizes compatibility between systems
  - Simplifies the design process
  - Promotes communication between people working on the system
  - Enables system refinement and adaptation through model transformation
Typical MDD steps

- **Green-field scenario:**
  1. Design a system using a widely-known modeling notation (preferably standard)
  2. Convert a model to an equivalent (as much as possible) model suitable for automated analysis
  3. Check the model (structural checks, simulation, model checking, theorem proving, etc.)
  4. Explain the results, locate problems on the model
  5. Generate executable code
  6. Generate tests and/or controls

- **Non-green field scenario**
  1. Extract a model (high-level or formalized) from executable specification

...
MDD with ECT

- Green-field scenario:
  1. Design a system using
     1. UML (Activity Diagrams or Sequence Diagrams)
        - Tool support: Eclipse UML Designer, BOUML
     2. Business Process Modeling Notation (BPMN) - Eclipse BPMN Designer
  2. Convert the model to Reo
  3. Check the Reo model
     1. Use animation and/or simulation plug-ins
     2. Use model checking tools
        1. Specify properties in LTL/ASL or μ-calculus
        2. Convert Reo to constraint automata or mCRL2 specification
        3. Verify using mCRL2 or Vereofy model checkers
  4. Show counterexamples on the Reo model
  5. Generate executable code

- Non-green field scenario
  1. Extract a Reo model from Business Process Execution Language (BPEL) specification

...
Why ECT?

- Good Petri net-based tools for business process analysis should deal with:
  - Extra arcs (e.g., reset and inhibitor)
  - Zero safe nets for modeling of distributed transactions
  - Colored Petri nets
  - Hierarchical Petri nets
  - Timed and stochastic Petri nets
  - ...

- ECT:
  - Propagation of synchrony is good for transaction modeling
  - More concise models of compensation and exception handing
  - Timed channels
  - Conversion of connectors to components and their reuse in higher-level models
  - Distinction between connectors (coordination code) and components (external functional code) helps to imitate the architecture of a service-based system
Agenda

- Model-driven development (MDD)
- Application of Eclipse Coordination Tools (ECT) to process verification
- **Formalization of process models (control flow)**
- Formalization of compliance requirements
- Modeling data flow
- Model checking data-aware Reo models with mCRL2
- Conclusions and Future Work
  - Models for QoS analysis
Formalization of business processes


Model checkers

- **Vereofy**
  - [http://www.vereofy.de/](http://www.vereofy.de/)
  - Developed at University of Dresden (SYANCO project)
  - Constraint-automata based symbolic model checker

- **mCRL2** behavioral specification language and associated toolset
  - [http://www.mcrl2.org/](http://www.mcrl2.org/)
  - Developed at TU Eindhoven
  - Based on the algebra of communicating processes (ACP)
  - Extended with data and time
  - Expressive property specification format (μ calculus)
  - Abstract data types, functional language (λ calculus)
  - Automated mapping from Reo to mCRL2
    - N. Kokash, E. d. V., C. Krause, Data-aware Design and Verification of Service Compositions with Reo and mCRL2, in: ACM Symposium on Applied Computing, 2010
mCRL2 specification language

- **Actions** are atomic events (e.g. a firing of a port or a request arrival in a Reo connector)
- **Processes** are the active entities defined as expressions over actions and other processes
  - Multiaction: $a/b$ (synchronized actions)
  - Alternative composition: $a + b$ (nondeterministic choice)
  - Sequence composition: $a.b$ ($b$ started after $a$)
  - Conditional: $exp \rightarrow a \diamond b$ (if-then-else)
  - At operator: $a^t$ (action $a$ happens at time $t$)
  - Parallel composition: $a || b$ (interleavings $a.b + b.a + a|b$)
  - Renaming: $\rho_R(a)$ where $R$ is a set of renamings of the form $b \rightarrow c$, meaning that every occurrence of $b$ in $a$ is replaced by $c$
  - Hiding: $\tau_H(a)$ renames all actions of $H$ in $a$ to $\tau$
  - Restriction (allow): $\nabla_R(a)$ where $R$ specifies which actions are allowed to occur in $a$
  - Blocking: $\partial_B(a)$ where $B$ is a set of actions that is not allowed to occur in $a$
  - Communication: $\Gamma_C(p)$, where $C$ is a set of allowed communications of the form $a_0|...|a_n \rightarrow c$, $n \geq 1$ which means that every group of actions $a_0|...|a_n$ within a multiaction is replaced by an action $c$

- **Actions and processes** can be parametrized with **data**
  - Summation: $\sum_{d \in D} a(d)$ ($a(d_1) + a(d_2) + a(d_3)\ldots$)
Example of mCRL2 specification

Dining philosophers

eqn $K = 2$;
map $K: \text{Pos}$;

act $\text{get, put, up, down, lock, free: } \text{Pos}#\text{Pos}$;
$\text{eat: } \text{Pos}$;

proc
$\Phi(n: \text{Pos}) = \text{get}(n, n) \cdot \text{get}(n, \text{if}(n == K, 1, n+1)). \text{eat}(n). \text{put}(n, n)$
$\cdot \text{put}(n, \text{if}(n == K, 1, n+1)) \cdot \Phi(n);$

$\text{Fork}(n: \text{Pos}) = \sum m: \text{Pos} \cdot \text{up}(m, n) \cdot \text{down}(m, n) \cdot \text{Fork}(n);$ 

init $\text{allow( } \{ \text{lock, free, eat} \}$,
$\text{comm( } \{ \text{get|up->lock, put|down->free} \}$,
$\Phi(1) \parallel ... \parallel \Phi(K) \parallel \text{Fork}(1) \parallel ... \parallel \text{Fork}(K))$;
Example of mCRL2 specification

Brief overview of the mCRL2 toolset:
• Process linearization – \texttt{mcrl2lps}
• Labelled Transition System (LTS) generation – \texttt{lps2lts}
• LTS minimization (e.g., branching bisimilarity) – \texttt{ltsconvert}
• LTS visualization – \texttt{ltsgraph}

• mCRL2 model checker
  • \texttt{lps2pbes} -formula lpsFile pbesFile
  • \texttt{pbes2bool} pbesFile
• Other tools (simulation, converters, etc.)

Dining philosophers
Data flow observed at a channel end = action

Synchronous channel, synchronous drain
- $Sync = A/B.Sync$;

Non-deterministic synchronous lossy channel
- $LossySync = (A/B + A).LossySync$;

Asynchronous drain
- $AsyncDrain = (A + B).AsyncDrain$;

FIFO1
- $FIFO1 = A.B.FIFO1$;
- $FullFIFO1 = B.FIFO1$;
- Alternative encoding: $FIFO1(b: \text{Bool}) = (\neg b \rightarrow A \Diamond B).FIFO1(\neg b)$;

Replication node
- $Replicator = X|Y|Z.Replicator$;

Merge node
- $Merger = (X/Z + Y/Z).Merger$;

Synchronize and hide actions corresponding to the connected channel
ECT: Generating mCRL2 specification for Reo

SYANCO winter school
Agenda

- Model-driven development (MDD)
- Application of Eclipse Coordination Tools (ECT) to process verification
- Formalization of process models (control flow)
- Formalization of compliance requirements
- Modeling data flow
- Model checking data-aware Reo models with mCRL2
- Conclusions and Future Work
  - Models for QoS analysis
Categories of compliance requirements
Managing compliance requirements

- Extract relevant requirements from legislation documents
  - Which documents/requirements are relevant for a given process?
  - Help of a domain expert is required
- Convert extracted requirements to formally specified properties
  - A technical expert is required
### Compliance Request (UI-110)

#### Select A Compliance Target
- Loan Processing
- Loan Originating
- Loan Monitoring
- Loan Closure
- Loan Development
- Sales Processing
- Purchase and Payables

#### Select A Compliance Source
- Basel II
- Sarbanes-Oxley
- SOX-404
- SOX-302
- ISO 17799
- ISO 17799 - 10.1.3
- Internal Policy - P

### Requirement - Risk - Control Profile

<table>
<thead>
<tr>
<th>C.Target</th>
<th>C.Requirement</th>
<th>C.Risk</th>
<th>Control</th>
<th>C.Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan Originating</td>
<td>Ensure customer is legally eligible for loan</td>
<td>Illegal loan processing</td>
<td>Customer's loan request is rejected if customer's banking privileges are suspended</td>
<td>Basel II Section 744</td>
</tr>
<tr>
<td>Loan Originating</td>
<td>Ensure loan is granted with adequate level of assurance</td>
<td>Lack of segregation of duties</td>
<td>Customer initial credit worthiness check is segregated from post check, which is performed by supervisor role</td>
<td>SOX Section 404, ISO 17799 - 10.1.3</td>
</tr>
<tr>
<td>Loan Originating</td>
<td>Ensure loan is granted with adequate level of assurance</td>
<td>Lack of independent management review</td>
<td>If the loan request is greater than 1M Euro, Loan contract is approved by the Manager otherwise it is approved by the Supervisor.</td>
<td>Bank Internal Policy, PS01, SOX Section 404</td>
</tr>
<tr>
<td>Loan Originating</td>
<td>Ensure adequate response time is provided to the customer</td>
<td>...</td>
<td>Bank waits for customer's sign for '10' days (legal delay) before it terminates the request</td>
<td>Directive No: 11</td>
</tr>
</tbody>
</table>

**Close**  **Check**
## Compliance management tools: requirement formalization

### Compliance Profile

<table>
<thead>
<tr>
<th>Compliance Target</th>
<th>Degree of Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan Origination (Definition 1)</td>
<td>100%</td>
</tr>
<tr>
<td>Loan Origination (Definition 2)</td>
<td>75%</td>
</tr>
</tbody>
</table>

### Compliance Requirement Details

<table>
<thead>
<tr>
<th>Compliance Requirement</th>
<th>Risk</th>
<th>Control</th>
<th>Source</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure customer is legally eligible for loan</td>
<td>Illegal loan processing</td>
<td>Customer’s loan request is rejected if customer’s banking privileges are suspended</td>
<td>Basel II Section 744</td>
<td>High</td>
</tr>
<tr>
<td>Ensure loan is granted with adequate level of assurance</td>
<td>Lack of segregation of duties</td>
<td>Customer initial credit worthiness check is segregated from post check, which is performed by supervisor role</td>
<td>SOX Section 404 ISO 17799 - 10.1.3</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>G(InitialCWC) (\land) (PostCWC)</td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
</tr>
<tr>
<td>2.2</td>
<td>G(PostCWC) (\rightarrow) G(PostCWC.Role(Supervisor))</td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
</tr>
<tr>
<td>2.3</td>
<td>G(InitialCWC) (\rightarrow) G(PostCWC.Role(Supervisor))</td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Activities performed by the same role</td>
</tr>
<tr>
<td>2.4</td>
<td>G(InitialCWC) (\land) G(PostCWC)</td>
<td>LTL</td>
<td>✓</td>
<td>?</td>
<td></td>
<td>This rule can be checked when Runtime information is available</td>
</tr>
</tbody>
</table>

| Ensure adequate response time is provided to the customer | Lack of independent management review | If the loan request is greater than 1M Euro, Loan contract is approved by the Manager otherwise it is approved by the Supervisor. | Bank Internal Policy PS01 SOX Section 404 | High |
| Ensure adequate response time is provided to the customer | ... | Bank waits for customer’s sign for 10 days (legal delay) before it terminates the request | Directive No: 11 | High |

### Loan Monitoring

*Loan Monitoring (Definition 1) (Last Update: 2009-05-01)*

*Loan Monitoring (Definition 2) (Last Update: 2009-05-02)*

100%
Compliance management tools:
Property template designer
µ-calculus

- Modal logic with fix points operators
  - Extends Hennessy-Milner Logic
  - Regular formulas (allow the use of sequences of actions in modalities)
  - Fixed point modalities
- Data and time

\[ \alpha ::= \tau a(t_1, ..., t_n) \alpha \]
\[ \alpha f ::= t | a | true | false | \alpha f \land \alpha f | \alpha f \lor \alpha f | \forall d : D . \alpha f | \exists d : D . \alpha f | af^c u \]
\[ R ::= \varepsilon | \alpha f \mid R \cdot R \mid R + R \mid R^* \mid R^+ \]
\[ \phi ::= true | false | \neg \phi | \phi \land \phi | \phi \lor \phi | \phi \rightarrow \phi | \forall d : D . \phi | \exists d : D . \phi | \langle R \rangle \phi | \phi [R] \phi \mid \Delta \Delta^c u \mid \nabla \nabla^c u \]
\[ \mu X (d_1 : D_1 := t_1, ..., d_n : D_n := t_n ). \phi | \nu X (d_1 : D_1 := t_1, ..., d_n : D_n := t_n ). \phi | X (t_1, ..., t_n) \]
Examples of compliance requirements expressed in μ-calculus

**Template:** If an action \( a \) happens, it is eventually followed by an action \( b \):

\[
[\text{true}.*.a.\text{true}.*]\langle\text{true}*.b.\text{true}.*\rangle\text{true}
\]

“If customer banking privileges are suspended a customer’s loan request is rejected”

\[
[\text{true}*.\text{privilegesSuspended.true}.*]\langle\text{true}*.\text{loanRequest.true}*.\text{reject.true}.*\rangle\text{true}
\]

**Template:** An action \( b \) happens within some time after an action \( a \):

“If a credit card fraud was discovered, the card should be blocked within ten minutes”

\[
[\text{true}*.\forall t : \mathbb{R}.\text{fraudDiscovered}^c t ]\langle\text{true}\rangle\exists u : \mathbb{R}.\left( u \leq t + 10 \land \text{suspendCard}^c u \right)
\]

**Template:** An action \( a \) must unavoidably be done:

“If a requested loan amount is higher than 10M, a manager authorization must be obtained”

\[
[\text{true}*.\exists \text{amount} : \mathbb{N}.\text{loanRequest(amount)} \land (\text{amount} > 1000000)] \mu X \overline{\text{authorization} X}
\]
Agenda

- Model-driven development (MDD)
- Application of Eclipse Coordination Tools (ECT) to process verification
- Formalization of process models (control flow)
- Formalization of compliance requirements
- **Modeling data flow**
- Model checking data-aware Reo models with mCRL2
- Conclusions and Future Work
  - Models for QoS analysis
Data-aware service coordination

CUSTOMER INTERFACE

HOTEL BOOKING SERVICE

SYANCO winter school
Data-aware service coordination


SYANCO winter school 26
Data-aware service coordination with Reo

<table>
<thead>
<tr>
<th>Data flow requirements language [MPT07]</th>
<th>Reo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic element</strong></td>
<td><strong>Graphical representation</strong></td>
</tr>
<tr>
<td>Node</td>
<td><img src="image" alt="Node symbol" /></td>
</tr>
<tr>
<td>Identity</td>
<td><img src="image" alt="Identity symbol" /></td>
</tr>
<tr>
<td>Fork</td>
<td><img src="image" alt="Fork symbol" /></td>
</tr>
<tr>
<td>Merge</td>
<td><img src="image" alt="Merge symbol" /></td>
</tr>
<tr>
<td>Cloner</td>
<td><img src="image" alt="Cloner symbol" /></td>
</tr>
<tr>
<td>Filter</td>
<td><img src="image" alt="Filter symbol" /></td>
</tr>
<tr>
<td>Last</td>
<td><img src="image" alt="Last symbol" /></td>
</tr>
<tr>
<td>Operation</td>
<td><img src="image" alt="Operation symbol" /></td>
</tr>
</tbody>
</table>

Data streams vs. timed data streams
Adding data to Reo

- **Writers/readers/components**
  - **Data domain:**
    \[ S = \text{struct } c_1(p_{11}:S_{11}, \ldots, p_{k1}:S_{k1}) \text{?} r_1 | \ldots | c_n(p_{1n}:S_{1n}, \ldots, p_{kn}:S_{kn}) \text{?} r_n \]
  - **Constraints**
    \[ \text{Writer} = \sum_{d \in \text{Nat}} (d < 10) \rightarrow A(d).\text{Writer} \]
- **Channels in a circuit must accept any data items**
  - **Data**
    \[ D = \text{struct } D_1(e_1 : DT_1) \text{?} isDT_1 | \ldots | D_n(e_n : DT_n) \text{?} isDT_n \]
  - **Join nodes:**
    \[ \text{Join} = \sum_{d_1, d_2 \in \text{Data}} (X(d_1) \cup Y(d_2) \cup Z(\text{tuple}(d_1, d_2))).\text{Join} \]
    \[ \text{Data} = \text{struct } D_1(e_1 : DT_1) \text{?} isDT_1 | \ldots | D_n(e_n : DT_n) \text{?} isDT_n | \text{tuple}(e_1 : \text{Data}, e_2 : \text{Data}) \text{?} \text{isTuple} \]
  - **Timers:**
    \[ \text{DataTimer} = \text{struct } \text{reset} \text{?} isReset | \text{off} \text{?} isOff | \text{timeout} | \text{other}(e : \text{Data}) \text{?} isOther \]
  - **Coloring semantics (to support context dependency):**
    \[ \text{Coloured} = \text{struct } \text{flow(data: Data)} | \text{noflowG} | \text{noflowR} \]
- **Filter conditions**
  - \( d : \text{Data} \)
  - \( (\text{isDT}_1(d) \rightarrow (e_1(d) == 0)) \rightarrow \ldots \)
- **Functions for data transformation**
  - \( f : \text{Data} \rightarrow \text{Data} \)
  - **Wrappers:** \( c : \text{InputDT} 
  \rightarrow \text{Data} \)
  - **Projections:** \( p : \text{Data} \rightarrow \text{OutputDT} \)
  - **Example:** \( \text{square}_\text{sum}(x,y) = z^2 \text{ whr } z = x+y \text{ end} ; \)
Reo with data to mCRL2

**act** A, B: Data

- Synchronous channel
  - \( \text{Sync} = \sum_{d \in \text{Data}} . A(d) | B(d) . \text{Sync} \)
- Synchronous drain
  - \( \text{SyncDrain} = \sum_{d_1,d_2 \in \text{Data}} . A(d_1) | B(d_2) . \text{SyncDrain} \)
- Synchronous lossy channel
  - \( \text{LossySync} = \sum_{d \in \text{Data}} . (A(d) | B(d) + A(d)) . \text{LossySync} \)
- Asynchronous drain
  - \( \text{AsyncDrain} = \sum_{d \in \text{Data}} . (A(d) + B(d)) . \text{AsyncDrain} \)
- Filter
  - \( \text{Filter} = \sum_{d \in \text{Data}} . (\text{exp}(d) \rightarrow A(d) | B(d) \diamond A(d)). \text{Filter}, \) where \( \text{exp}(d) \) is a boolean expression
- Transformer
  - \( \text{Transformer} = \sum_{d \in \text{Data}} . A(d) | B(f(d)) . \text{Transformer} \)
- Replication node
  - \( \text{Replicator} = \sum_{d \in \text{Data}} . X(d) | Y(d) | Z(d) . \text{Replicator} \)
- Merge node
  - \( \text{Merger} = \sum_{d \in \text{Data}} . (X(d) | Z(d) + Y(d) | Z(d)) . \text{Merger} \)
- FIFO1
  - \( \text{DataFIFO1} = \text{struct} \) empty?isEmpty | full(e:Data)?isFull;
  - \( \text{FIFO1}(f: \text{DataFIFO1}) = \sum_{d \in \text{Data}} \text{isEmpty}(f) \rightarrow A(d).\text{Fifo1}(\text{full}(d)) \diamond B(e(f)).\text{FIFO1}(\text{empty})) \)
Example (Check fragment)

struct el(activated: Bool, amount: Nat)

(amount(d)<1) No data
Example (Auction)

- **DataWriter1 = struct auctionInfo**
  - auctionID: Pos,
  - auctionOpen: Bool,
  - sellerID: Nat,
  - startPrice: Nat,
  - bidderID: Nat,
  - bidderPrice: Nat)?isAuctionInfo;

- **DataWriter2= struct bidInfo**
  - newBidderID: Nat,
  - newBidderPrice: Nat)?isBidInfo;

- **Data = struct d1**
  - e1 : DataWriter1)?isDataWriter1 |
  - d2(e2: DataWriter2)?isDataWriter2
Variable = Write.(Write + Read).Variable

\[
\text{Variable}(f: \text{DataFIFO}) = \sum_{d \in \text{Data}}( \\
  (f == \text{empty}) \rightarrow \\
  \text{Write}(d).\text{Variable} (\text{full}(d))) \\
\diamond (\text{Write}(d).\text{Variable}(\text{full}(d)) + \\
  \text{Read}(e(f)).\text{Variable}(f))
\]

Data = struct a|b
Agenda

- Model-driven development (MDD)
- Application of Eclipse Coordination Tools (ECT) to process verification
- Formalization of process models (control flow)
- Formalization of compliance requirements
- Modeling data flow
- Model checking data-aware Reo models with mCRL2
- Conclusions and Future Work
  - Models for QoS analysis
A problem with constraint propagation:

- $DT_1 = \text{struct } d_1(a: \text{Nat}), DT_2 = \text{struct } d_2(b: \text{Nat})$
- $Writer1 = \sum_{d_1 \in DT_1} (a(d_1) == 10) \rightarrow A(d_1)$
- $\sum_{d_1 \in DT_1, d_2 \in DT_2} \cdot (a(d_1) == b(d_2)) \rightarrow B(d_1) | C(d_2)$
- $\text{comm}(A|B \rightarrow X)$

- $\text{DataWriter1} = \text{struct bidInfo(}
  \text{newBidderID: Nat,}
  \text{newBidderPrice: Nat)?isBidInfo;}
- $Writer1 = \sum_{d \in Data} ((\text{bidderID}(d) == 1) \& \& (\text{bidderPrice}(d) == 10))…$
Example (Auction, no data)
Example (Auction, with data)
Example (Auction, with data and colors)
Agenda

- Model-driven development (MDD)
- Application of Eclipse Coordination Tools (ECT) to process verification
- Formalization of process models (control flow)
- Formalization of compliance requirements
- Modeling data flow
- Model checking data-aware Reo models with mCRL2
- Example
- Conclusions and Future Work
  - Models for QoS analysis
Future Work

- Work on tools usability
  - Analysis across several connectors
  - User friendly environment for data annotation
  - Import mCRL2 component specifications
  - Libraries of useful components
  - Better verification algorithms
  - Integration with service interface description standards
  - Automatic service binding

- Models and tools for QoS analysis
  - Observe different events (actions) on Reo ports (e.g., “port A is ready to accept data”, “port A is accepting data”, “port A finished a data transfer”)

SYANCO winter school
Synchronous transactions with explicit data flow modeling

- \( t_2 \gg t_1 \)
- This automaton does not allow multiple transitions \( \{A,B\} \) while C is reading data.

Data transfer delay: 
\[
\max(t_1, t_2) + \max(t_3, t_4, t_5) \text{ or } \max(t_1 + \max(t_3, t_4), t_2 + \max(t_3, t_5))
\]
Synchronous transactions with explicit data flow modeling

- Actions observed in a port A:
  - **block** (bA)
  - **start** data flow (sA)
  - **finish** data flow (fA)
  - **unblock** (uA)
  - Port behavior: bA.sA.fA.uA

- Synchronous channel:
  - bA/bB.sA/sB.fA/fB.uA/uB

- LossySync:
  - bA/bB.sA/sB.fA/fB.uA/uB + bA.sB.fB.uB

- FIFO1 (empty):
  - bA.sA.fA.uA.bB.bB.sB.fB.uB

- SyncDrain:
  - bA/bB.(sA.fA || sB.fB).uA/uB

- AsyncDrain:
  - bA.sA.fA.uA + bB.sB.fB.uB
Synchronous transactions
with explicit data flow modeling

- **Block** and **unblock** actions are synchronized as in constraint automata
- **Source/source**
  - $sA|sB$
- **Sink/Source**
  - $fA|sB$
- **Sink/sink**
  - Merge node
    - $bX|bZ.sX|sZ.fX|fZ.uX|uZ + bY|bZ.sY|sZ.fY|fZ.uY|uZ$
- **Performance analysis:**
  - Port $A$ pumps data = state between $sA$ and $fA$
  - Time delays can be assigned to “finish” actions
LTS with data flow
Conclusions

- ECT is a suitable environment for business process analysis
  - Model-driven development
  - Fits service-oriented computing paradigm
  - Automated conversion
  - Formal models
  - Data flow model checking
    - Possible!
    - Quite expressive
    - Difficult to use
- Tools and models for QoS analysis are needed
  - mCRL2 is an easy way to test new semantics for Reo