Mining Workflow Recovery From Event Based Logs

Walid Gaaloul, *LORIA (INRIA - CNRS - Universities of Nancy), France*

Claude Godart, *LORIA (INRIA - CNRS - Universities of Nancy), France*

BPM'2005, Nancy, France

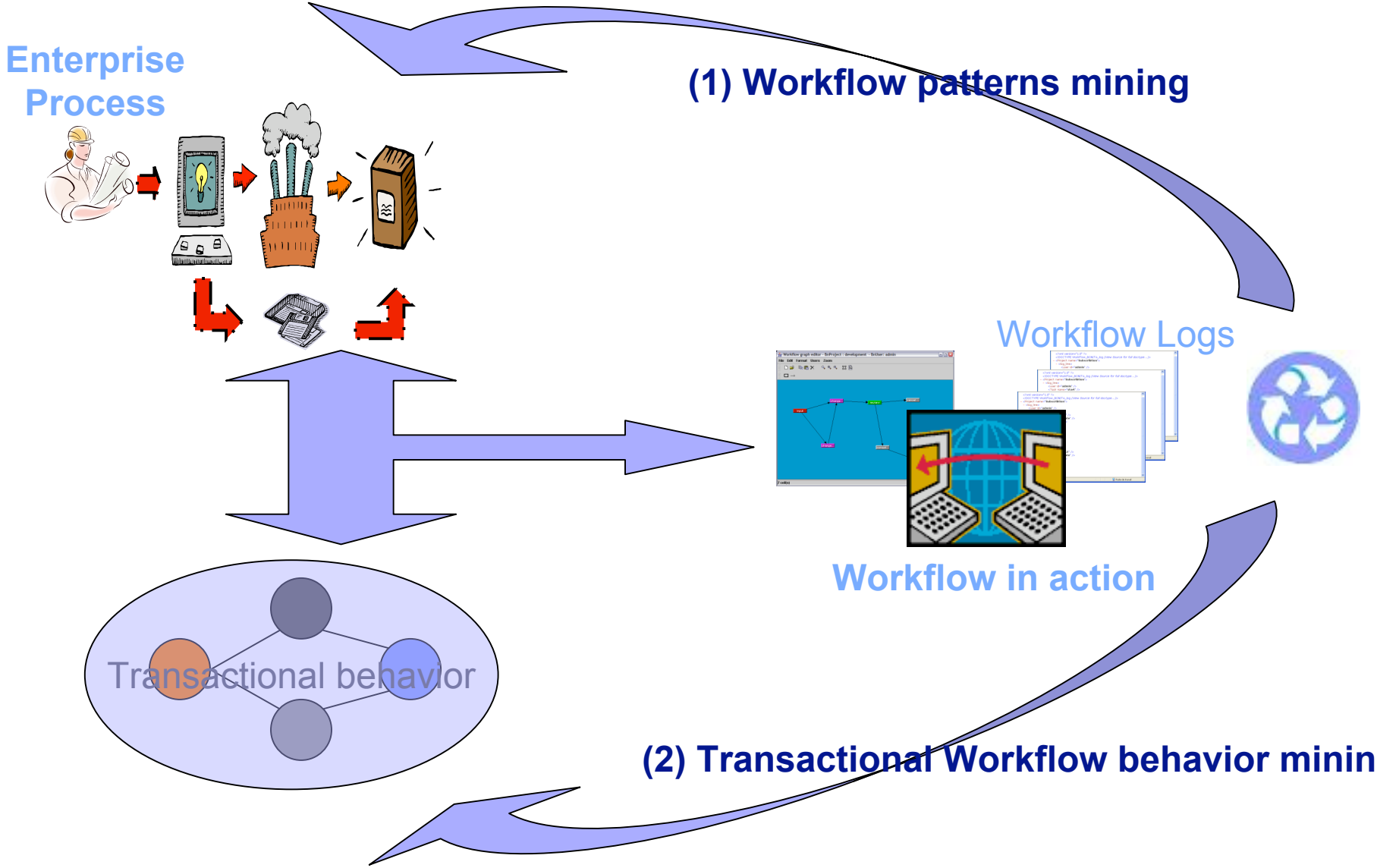
Outlines

- Prerequisites
- Mining workflow patterns
- Mining transactional behavior
- Improving workflow recovery
- Conclusion

Recovery mechanisms and transactional workflows

- Recovery mechanisms :
  - are applied in case of failures to resume correctly workflow execution from a consistent point;
  - use transactional mechanisms to guarantee reliable and consistent executions.
- Transactional workflows :
  - focus on issues of consistency from the business point of view rather than from the database point of view;
  - add additional mechanisms to support the automation of failure handling during runtime.

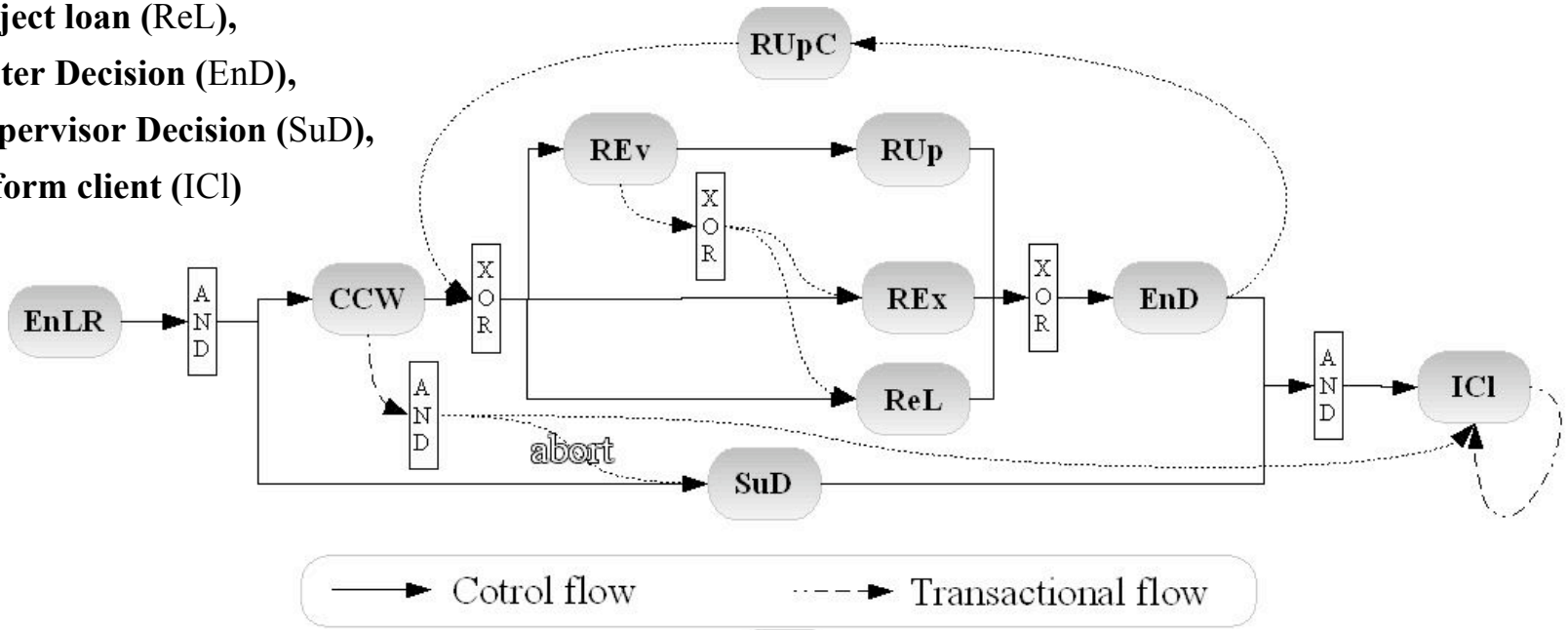
Scope and objectives



Motivating Example

- **Enter loan terms** (EnLR),
- **Client credit worthness** (CCW),
- **Risk evolution** (REv), **Risk update** (RUp)
- **Risk exception** (REx),
- **Reject loan** (ReL),
- **Enter Decision** (EnD),
- **Supervisor Decision** (SuD),
- **Inform client** (ICl)

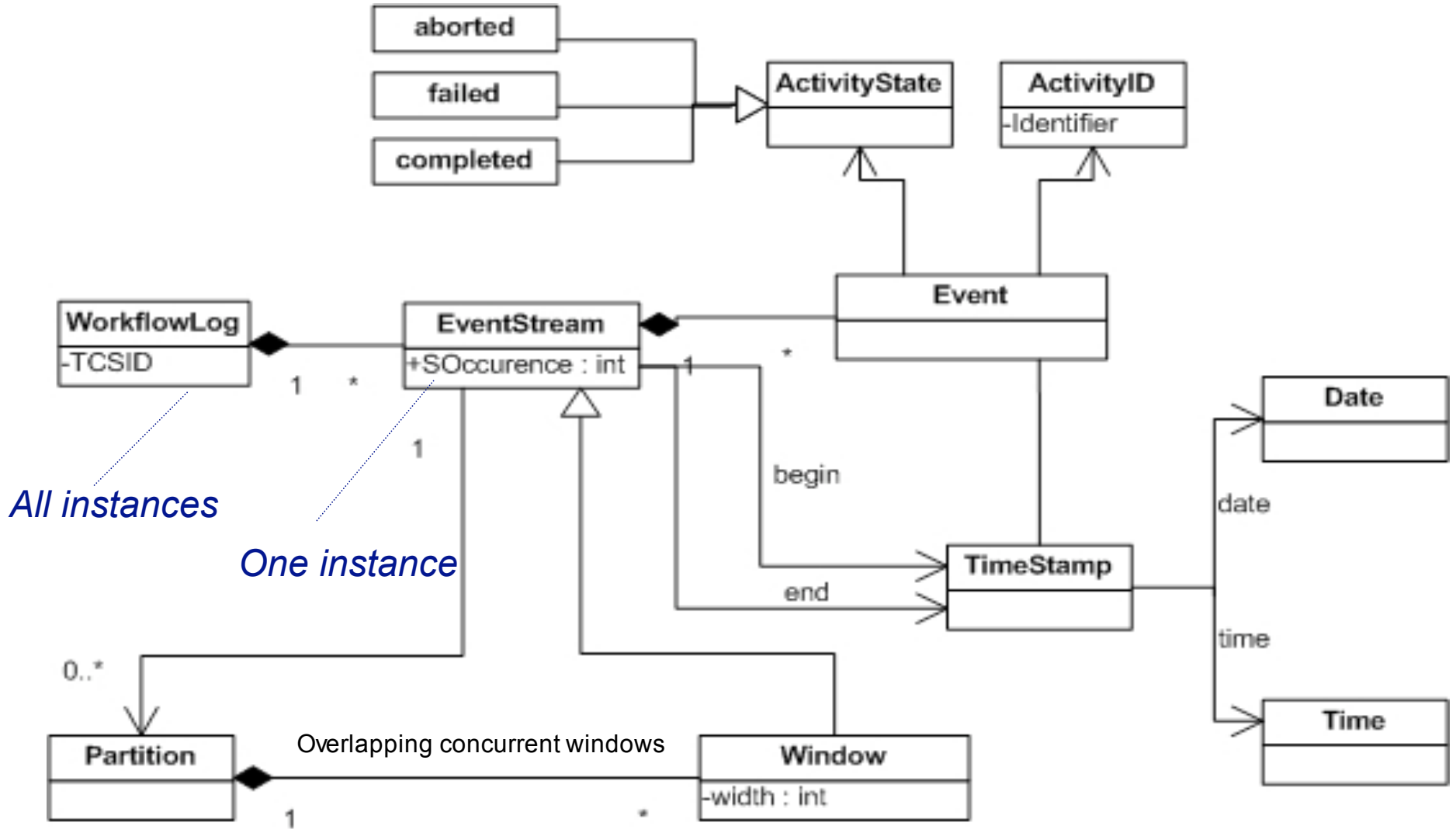
Additional transactional behavior specifies recovery mechanisms



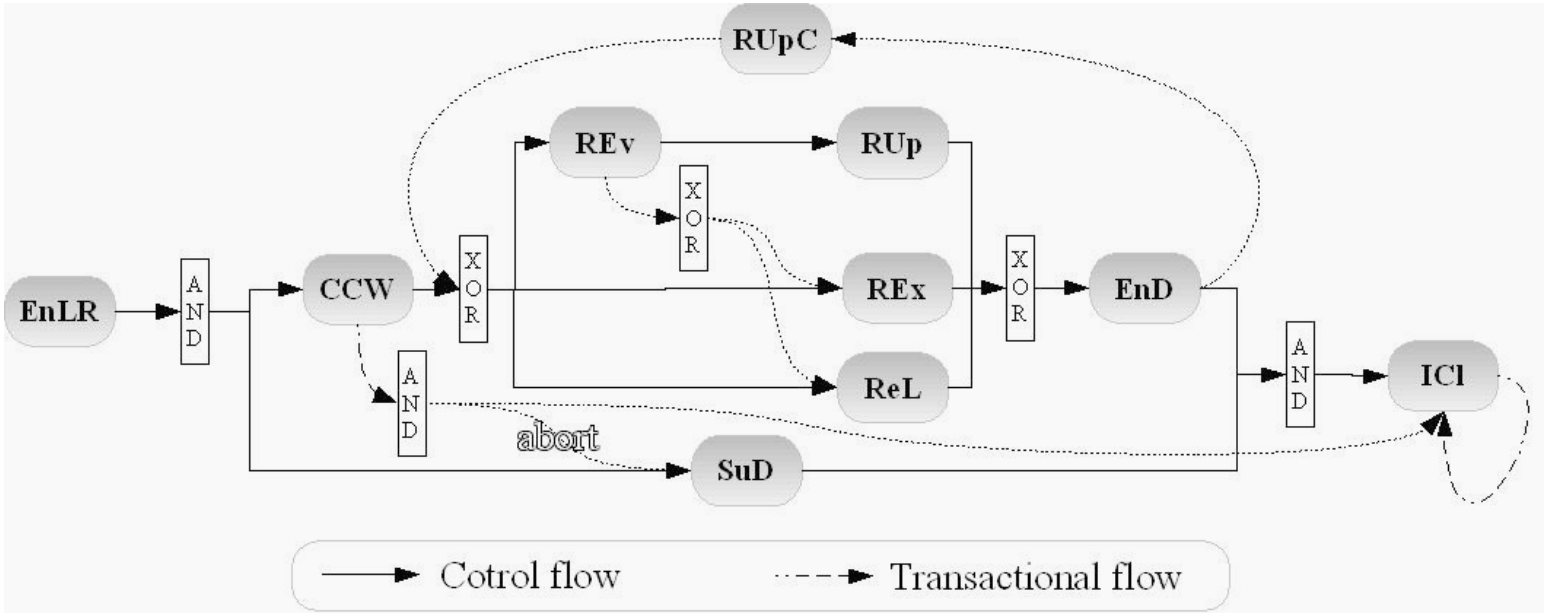
- *CCW* never fails
- *SuD* can fail

- No need to specify a recovery mechanism for CCW
- Abort concurrent activities of decision process and resume workflow execution

Workflow Log Model

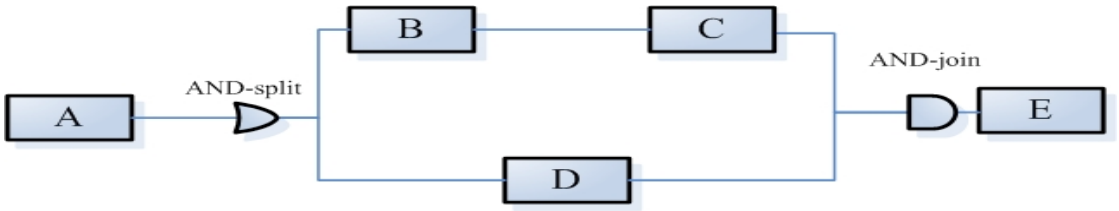


A Workflow example and one of its instances Event Stream



EventStream(5, 13, [**event**(EnLR,5, terminated), **event**(CCW,7,terminated), **event**(REv,7,failed), **event**(SuD,10,terminated), **event**(REx,11, terminated), **event**(End,12, terminated), **event**(ICl,13, failed), **event**(ICl,13, terminated)])

Log Analysis



- For each activity A_i in workflow log :
  - $\#A_i$: the overall occurrence of A_i
  - $p(A_i \mid A_j)$: the dependency of A_i to a "directly" previous activity A_j

instance 1 : ABDCE
instance 2 : ADBCE
instance 3 : ABCDE

e.g. $p(B \mid A) = 2 / 3$

#A = #B = #C = #D = #E = 3

	A	B	C	D	E
A	0.0	0.0	0.0	0.0	0.0
B	2/3= 0.66	0.0	0.0	1/3 = 0.33	0.0
C	0.0	2/3 = 0.66	0.0	1/3 = 0.33	0.0
D	1/3= 0.33	1/3 = 0.33	1/3 = 0.33	0.0	0.0
E	0.0	0.0	2/3 = 0.66	1/3 = 0.33	0.0

Log Analysis

- Problems :
  - Erroneous Dependencies
  - Undetectable Dependencies
  - More details are given in our papers in CoopIS'04 or DEXA'05 conferences
- Statistical properties :
  - P1: Mutual exclusive dependency property
  - P2: Concurrency property :
    - Global concurrency ,Partial concurrency, No concurrency.
  - P3: Choice property:
    - Free choice, Single choice, No choice

Patterns rules

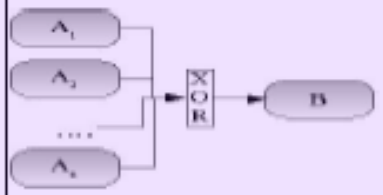
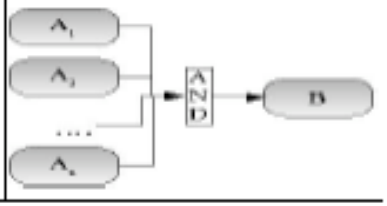
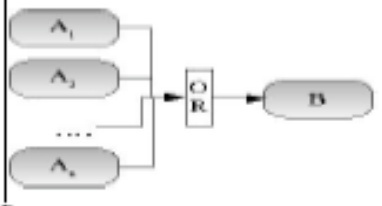
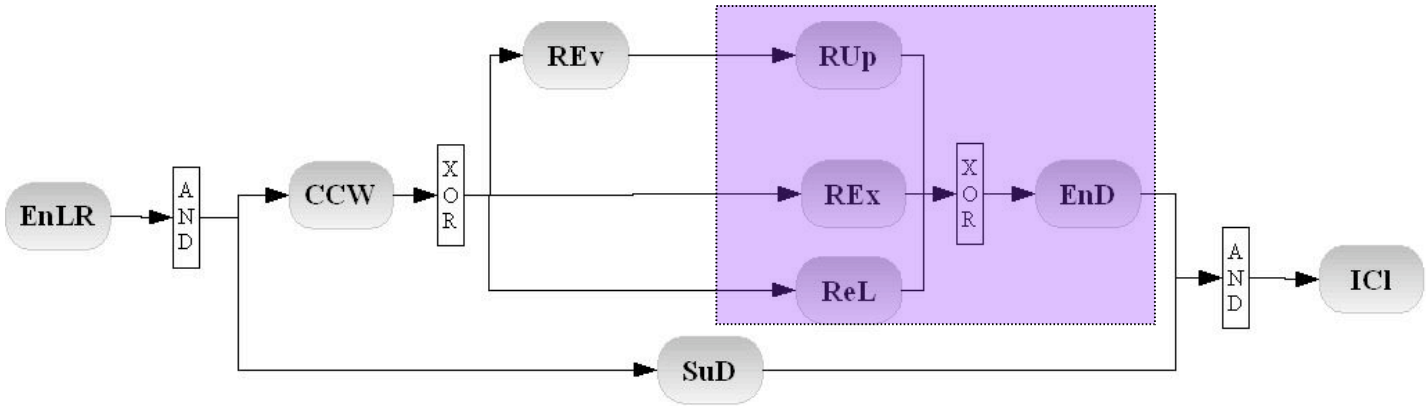
Rules	workflow patterns
(P3)($\Sigma_{i=0}^{n}$ (#A_i)=#B)	xor-join pattern
(P3)($\Sigma_{i=0}^{n}$ P(B/A_i)=1) $\wedge$ (P2)($\forall 0 \leq i,j \leq n; P(A_i/A_j) = 0$)	
(P3)($\forall 0 \leq i \leq n$; #A_i=#B)	and-join pattern
(P3)($\forall 0 \leq i \leq n; P(B/A_i) = 1$)$\wedge$ (P2)($\forall 0 \leq i,j \leq n\ P(A_i/A_j) = -1$)	
(P3)($m * \#B \leq \Sigma_{i=0}^{n}$ (#A_i)) $\wedge$ ($\forall 0 \leq i \leq n$; #$A_i \leq$ #B)	M-out-of-N-Join pattern
(P3)($m \leq \Sigma_{i=0}^{n} P(B/A_i) \leq n$) $\wedge$ (P2)($\exists 0 \leq i,j \leq n; P(A_i/A_j) = -1$)	

Table 4. Rules of join workflow patterns

Patterns rules

P(x,y)	EnLR	CCW	REv	RUp	REx	ReL	EnD	SuD
EnLR	0	0	0	0	0	0	0	0
CCW	1	0	0	0	0	0	0	-1
REv	0	1	0	0	0	0	0	-1
RUp	0	0	1	0	0	0	0	-1
REx	0	1	0	0	0	0	0	-1
ReL	0	1	0	0	0	0	0	-1
EnD	0	0	0	0.58	0.23	0.19	0	-1
SuD	1	-1	-1	-1	-1	-1	-1	0

$$EnLR = CCW = EnD = SuD = 100$$
$$REv = RUp = 69 REx = 21 ReL = 10$$

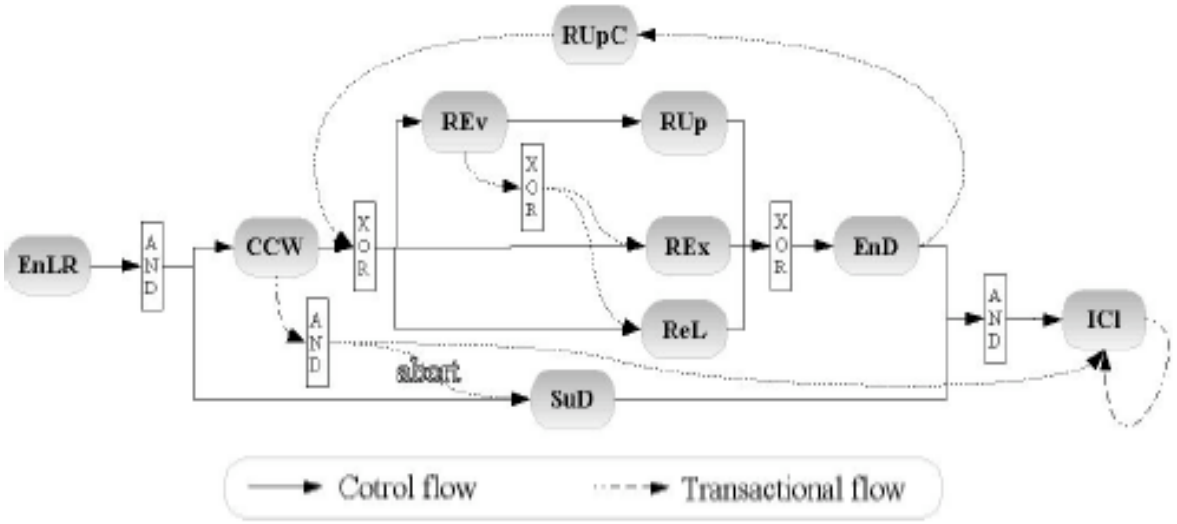


Transactional Workflows

- Control flow : workflow patterns [van der Aalst]
- Transactional behavior :
  - Activities transactional properties :
    - Depending on transitions of internal states (aborted, failed and completed);
    - Transactional properties : retriable, compensatable, pivot [Elmagarmid].
  - Transactional flow :
    - External transitions ensuring failure handling by resuming workflow execution from a consistent point
    - Consistent point represents an acceptable intermediate execution state from a business perspective where certain actions can be taken to fix the problem that caused the failure or choose an alternative execution path

Mining inter activities transactional dependencies

- ITR_A(e1, e2) table reports inter activities transactional dependencies after activity ”A” failures where :
  - (ei.state= failed and ei.activity = ”a”; i=1,2) OR
  - ei.activity; is a ”new” activity (i.e not appear in control flow mining); i=1,2)



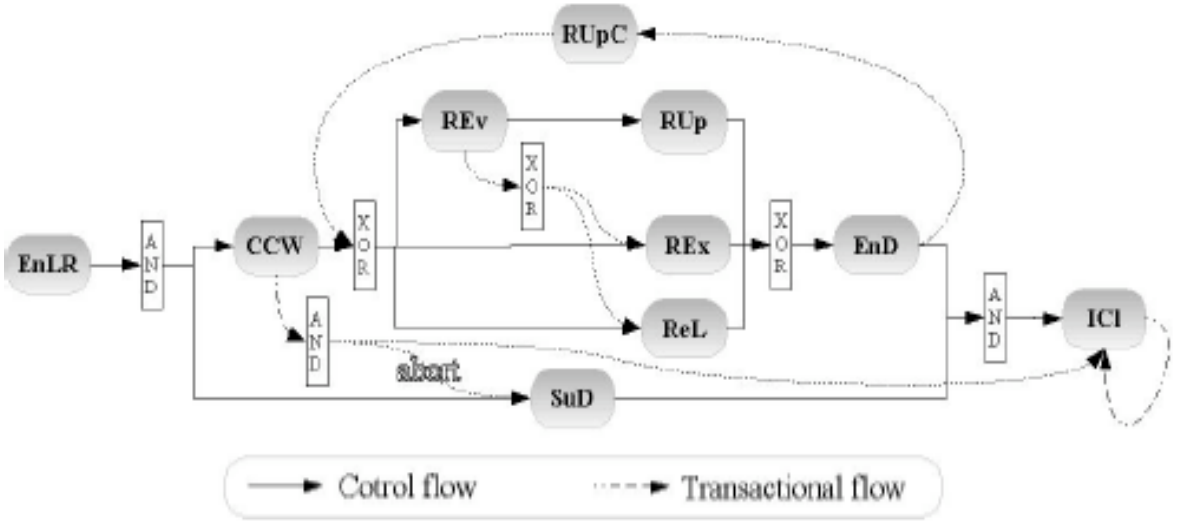
ITR_{EnD}	EnD,f	RUpC,c	REv,c	REx,c	ReL,c	RUp,c
EnD,f	0	0	0	0.56	0.34	0.1
RUpC,c	1	0	0	0	0	0
REv,c	0	1				
REx,c	0	1				
ReL,c	0	1				
RUp,c	0	0				

Mining transactional flow

- Alternative Dependency:
  - ITR_{a2}((“a1”, completed), (“a2”, failed)) >0;
  - We distinguished two kind of alternative dependencies:
    - a backward alternative : if the consistent state is located before the failed activity
    - a forward alternative : if the consistent state is located after the failed activity
- Abortion Dependency:
  - ITR_{a2}((“a1”, aborted), (“a2”, failed)) >0

Mining intra activities transactional dependencies

- ATR^{A}_{act} intra activities transactional dependencies table
  - Reports the intra states transitions of the ”A” activity after ”act”’s failures.
  - These transitions are extracted from a workflow logs projection that take only events of ”A” after ”act”’s failures.



ATR^{REv}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

ATR^{REx}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

ATR^{RUp}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

ATR^{ReL}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

Mining activities transactional properties

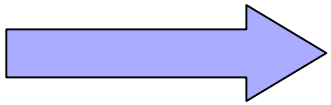
- An activity “a” is said to be retriable *iff* ATR^{a}_{a}(completed, failed)=1 Λ ITR_{a}((“a”, completed), (“a”, failed)) = 1.
- An activity “a” is said to be pivot *iff* √”act” ATR^{a}_{act}(”x”, c) = 0; *(”x”= completed or aborted or failed.)*

ATR^{REv}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

ATR^{REx}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

ATR^{RUp}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0

ATR^{ReL}_{End}	c	f	a
c	1	0	0
f	0	0	0
a	0	0	0



REv, Rex, RUp and ReL are not pivot

Improving workflow recovery

- Requirements :
  - Workflow recovery depend on the semantics of the process ;
  - Minimize the amount of effort resuming the workflow execution;
  - Specify only to needed “failed activity” an “optimal” recovery mechanisms to resume workflow execution.
- How :
  - Correct or remove the wrong transactional behavior;
  - Suggest relevant transactional behavior for a better failure handling and recovery.

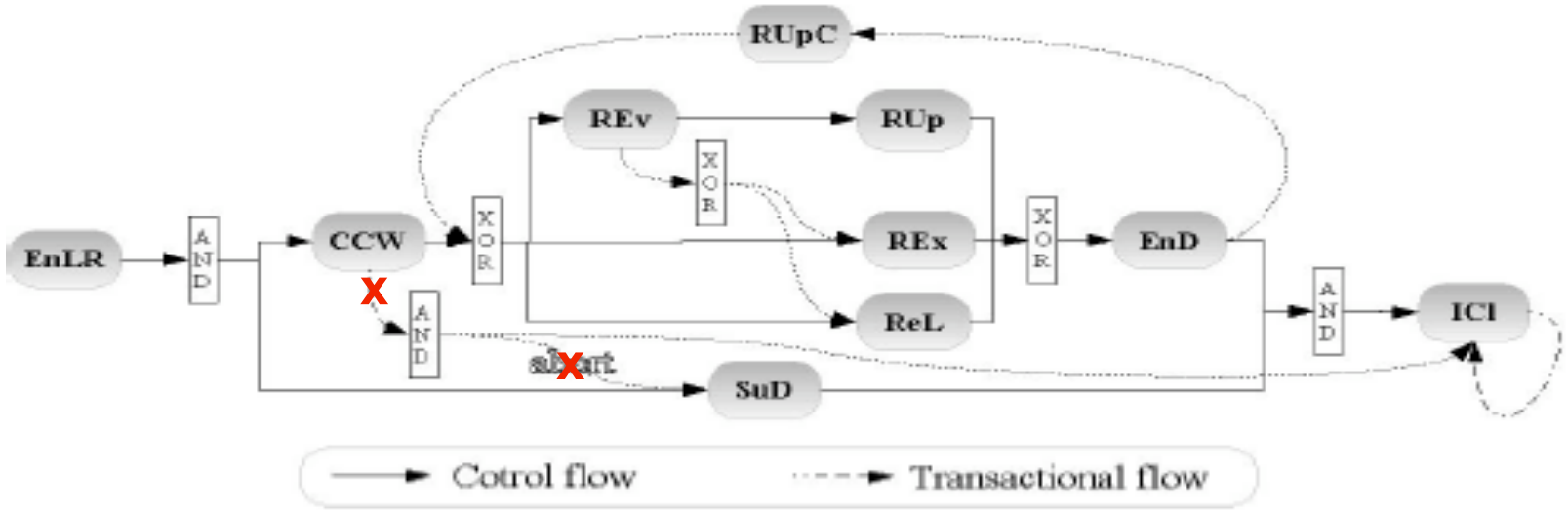
Improving workflow recovery

- Vital activity :
  - if a vital activity fails then it must be recoverable
  - All activities are vital for a workflow execution except in some parallel threads of M-out-of-N Join and OR-Join patterns;
- For XOR-split, XOR-join, sequence patterns : There is no abortion dependencies between the activities. An abortion dependency can exist only in case of failure and between parallel activities;
- For AND-split, AND-join patterns combination : If a vital activity fails then the consistent point must be out of the parallel threads between the ”join” and ”fork” point;
- For all workflow patterns : If we have a backward recovery then the consistent point cannot be before an executed pivot activity.

Improving workflow recovery

We discovered that CCW never fails:

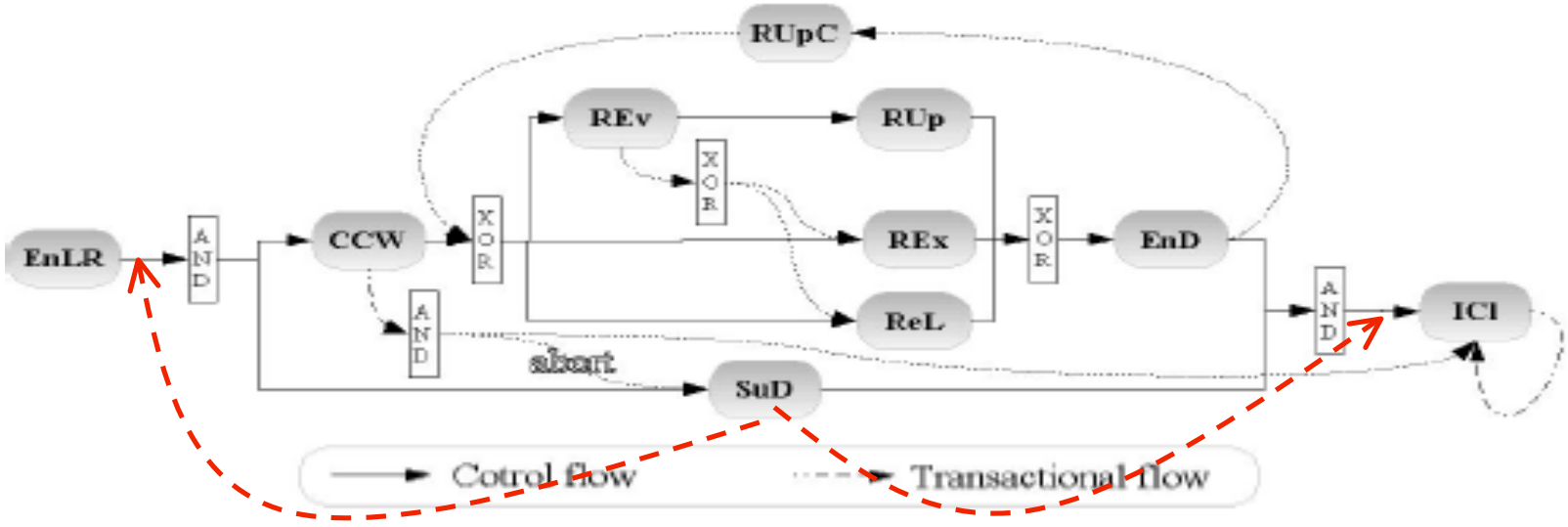
- By applying R1 there is no need for CCW to be recoverable.
- By applying R2 we remove the abortion dependency between CCW and SuD.



Improving workflow recovery

We discovered that SuD can fail :

- By applying R3 the consistent point after SuD failure is localized just before ICI or just after EnLR;
- By applying R4 we can only suggest a forward recovery, after SuD failure, if EnD or CCW are pivot activities;
- By applying R2 we can suggest to specify abortion dependencies between SuD in one side and (CCW, REv, RUp, REx and ReL) activities in another side.



Conclusion

- Our workflow patterns mining approach
  - Deals dynamically with concurrent behavior
  - Proceeds through a local log analyzing that allow us to recover partial results.
- Our mining transactional behavior approach
  - Discovers workflow transactional behavior
  - Identifies workflow recovery techniques
  - improves and corrects workflow failure handling using mining techniques

Perspectives

- Workflow mining:
  - Validate and test our approach on real workflow applications *(WorkflowMiner over BONITA WfMS)*;
  - Discover more complex patterns by using more metrics;
  - Discover more complex transactional characteristics.
- Applying existing mining techniques to other problems
  - Web service mining (*in progress*)

Thank You
for your attention