Architectures for a Temporal Workflow Management System

Carlo Combi
Dipartimento di Informatica
Università di Verona, strada le Grazie 15
37134 Verona, Italy
carlo.combi@univr.it

Giuseppe Pozzi
Dipartimento di Elettronica e Informazione
Politecnico di Milano, P.za L. da Vinci 32
20133 Milano, Italy
giuseppe.pozzi@polimi.it

Talk Overview

- Introduction;
- Workflow management systems: basic concepts;
- Managing temporal aspects in workflow systems;
- Different architectures for a temporal WfMS;
- Implementation issues;
- Conclusions and further work.

Introduction

- Any business process requires the coordinated execution of single activities to achieve a common goal: a workflow formally describes these activities, including criteria to assign single activities to executing units.
- Workflow management systems, *WfMS*, are software systems supporting the execution of workflow instances. Most WfMSs use a database management system (DBMS) based on the relational model.
- Considerable advantages may be obtained by the adoption of a temporal architecture for the WfMS. Unfortunately, due to the lack of a high-performance and reliable temporal database systems, no WfMS exists relying on top of a temporal database system.

Goal of the work

- Discuss *different architectures for a temporal WfMS*;
- Deal with some *implementation issues*.

WfMS: basic concepts

The data models defined in a WfMS are related to:

- the organization where the business process is enacted (*organizational* model),
- the structure of the managed business process (*process* model),
- the application specific information (*information* model).

WfMS data models and temporal information

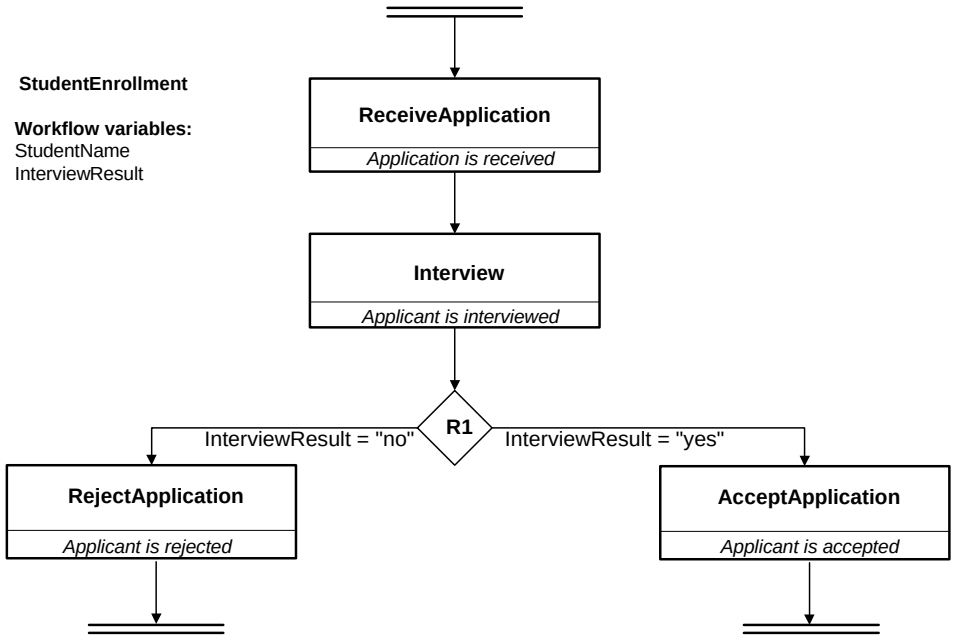
- **Organizational model**. By suitable tables the organizational model describes the agents and their availability time. Temporal aspects are mainly related to working days and hire time of the agent.
- **Process model**. The process model describes the schema of the managed business process. It is defined by predecessor-successor relationships and may also include routing tasks (*split* and *join*). Temporal aspects are related to changes of a schema of a business process.
- **Information model**. The information model considers both process specific and historical data. The first ones are data collected by the case (i.e., a process instance). The latter ones describe the history of the cases managed by the WfMS.

A concrete situation

The enrollment process of graduated students applying for PhD candidate position: starting from September 1^{st}, 2003, any received application leads to an interview of the applicant. After a few days, the university realizes that interviewing any student is extremely expensive. After September 30^{th}, 2003, the new adopted process model states that applicants' CVs are analyzed first: applicants whose CV is passed will be interviewed, only.

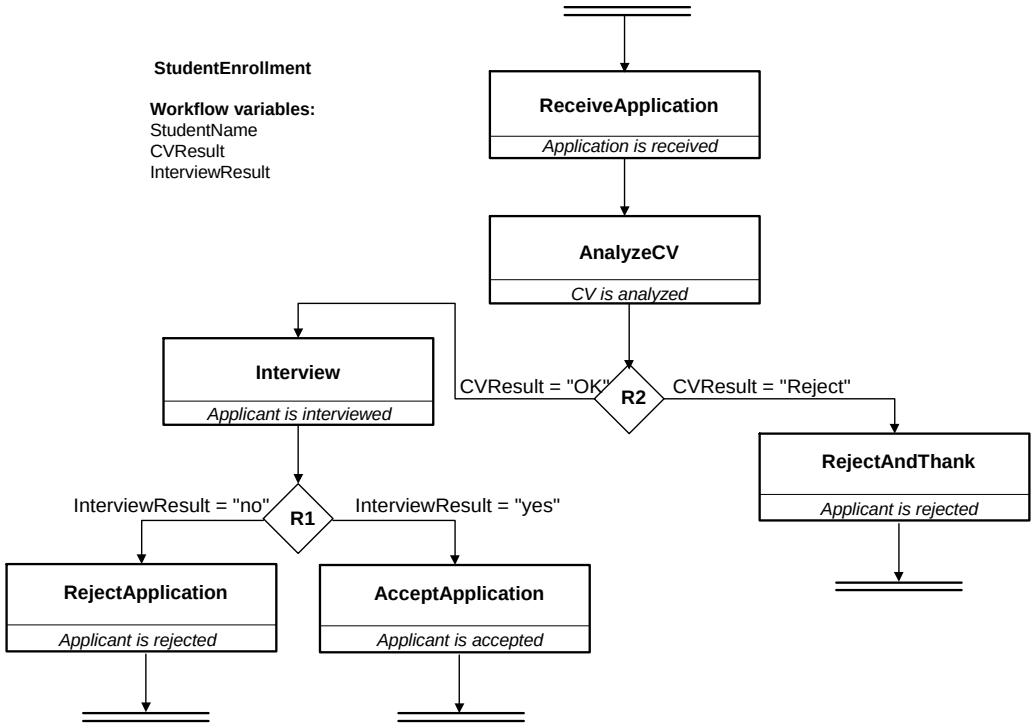
A concrete situation

The enrollment process: first version



A concrete situation

The enrollment process: second version



Temporal tables for the organizational model

Agent	AgentId	Role	Availability	VT
	H03	CommitteeMember	working days in [00-08-09 $\div +\infty$]	[00-08-09 $\div +\infty$]
	R01	Secretary	Monday, Tuesday in [03-01-05 $\div +\infty$]	[03-01-05 $\div +\infty$]
	L05	CommitteeMember	working mornings in [03-08-01 $\div +\infty$]	[03-08-01 $\div +\infty$]
	K13	CommitteeMemeber	working days in [03-01-01 $\div +\infty$]	[03-01-01 $\div +\infty$]
	D10	CommitteePresident	working days in [01-01-01 $\div +\infty$]	[01-01-01 $\div +\infty$]
	B08	CommitteeMember	working days in [01-11-12 $\div +\infty$]	[01-11-12 $\div +\infty$]
	B01	Secretary	Wednesday, Thursday, Friday in [00-01-03 $\div +\infty$]	[00-01-03 $\div +\infty$]

Temporal tables for the process model

WorkFlow	SchemaName	StartTask	VT
	StudentEnrollment	ReceiveApplication	[03-09-01 $\div$ $+\infty$]

WorkTask	SchemaName	TaskName	Role	VT
	StudentEnrollment	ReceiveApplication	Secretary	[03-09-01 $\div$ $+\infty$]
	StudentEnrollment	Interview	CommitteeMember	[03-09-01 $\div$ $+\infty$]
	StudentEnrollment	RejectApplication	CommitteePresident	[03-09-01 $\div$ $+\infty$]
	StudentEnrollment	AcceptApplication	CommitteePresident	[03-09-01 $\div$ $+\infty$]
	StudentEnrollment	AnalyzeCV	CommitteeMember	[03-10-01 $\div$ $+\infty$]
	StudentEnrollment	RejectAndThank	CommitteePresident	[03-10-01 $\div$ $+\infty$]

Temporal tables for the process model

Next	SchemaName	TaskName	NextTask	Tcons	VT
	StudentEnrollment	ReceiveApplication	Interview	[0 m ÷ 2 m]	[03-09-01 ÷ 03-09-30]
	StudentEnrollment	Interview	R1	[0 m ÷ 10 m]	[03-09-01 ÷ +∞]
	StudentEnrollment	RejectApplication	end_flow	[0 m ÷ 1 m]	[03-09-01 ÷ +∞]
	StudentEnrollment	AcceptApplication	end_flow	[0 m ÷ 0 m]	[03-09-01 ÷ +∞]
	StudentEnrollment	RejectAndThank	end_flow	[0 m ÷ 0 m]	[03-10-01 ÷ +∞]
	StudentEnrollment	ReceiveApplication	AnalyzeCV	[0 m ÷ 0 m]	[03-10-01 ÷ +∞]
	StudentEnrollment	AnalyzeCV	R2	[0 m ÷ 2 m]	[03-10-01 ÷ +∞]

RoutingTask	SchemaName	RTName	Type	VT
	StudentEnrollment	R1	mutualex_fork	[03-09-01 ÷ +∞]
	StudentEnrollment	R2	mutualex_fork	[03-10-01 ÷ +∞]

AfterFork	SchemaName	ForkTask	NextTask	Cond	VT
	StudentEnrollment	R1	RejectApplication	InterviewResult = “no”	[03-09-01 ÷ +∞]
	StudentEnrollment	R1	AcceptApplication	InterviewResult = “yes”	[03-09-01 ÷ +∞]
	StudentEnrollment	R2	Interview	CVResult = “OK”	[03-10-01 ÷ +∞]
	StudentEnrollment	R2	RejectAndThank	CVResult = “Reject”	[03-10-01 ÷ +∞]

Temporal tables for the information model

StudentEnrollmentData	CaseId	StudentName	CV Result	InterviewResult	VT
	27	Marple T.R.	n/a	"yes"	[03-09-09 ÷ +∞]
	89	Wallace E.S.	"yes"	"no"	[03-10-03 ÷ +∞]

CaseHistory	CaseId	SchemaName	Responsible	VT
	27	StudentEnrollment	R01	[03-09-09 10:03:10 ÷ 03-09-27 13:25:50]
	89	StudentEnrollment	B01	[03-10-03 9:00:10 ÷ 03-10-31 10:19:09]

TaskHistory	CaseId	TaskName	FinalState	Agent	VT
	27	ReceiveApplication	Completed	R01	[03-09-09 10:03:10 ÷ 03-09-09 10:05:50]
	27	Interview	Completed	H03	[03-09-27 12:06:00 ÷ 03-09-27 12:25:50]
	27	AcceptApplication	Completed	D10	[03-09-27 13:20:51 ÷ 03-09-27 13:25:50]
	89	ReceiveApplication	Completed	B01	[03-10-03 9:00:10 ÷ 03-10-03 9:07:43]
	89	AnalyzeCV	Completed	L05	[03-10-10 16:01:45 ÷ 03-10-10 16:12:41]
	89	Interview	Completed	B08	[03-10-31 9:32:03 ÷ 03-10-31 10:03:07]
	89	RejectApplication	Completed	D10	[03-10-31 10:03:10 ÷ 03-10-31 10:19:09]

Workflow Components

Workflow Management Systems (WfMSs) are complex software systems and include many different components, such as:

- the *workflow engine*, made of the interpreter of the process definition language (PDL) and of the workflow scheduler;
- the process model designer unit, which helps the workflow designer to suitably define a process model according to the supported PDL;
- the resource management unit (also known as *resource executive*), to assign tasks to executing agents;
- the database connectivity unit, to access data stored into a DBMS;
- the transaction manager;
- the e-mail feeder, to send agents messages and attached documents;
- the web server and the worklist server.

Workflow Components and Temporalities

- The *workflow engine* reads the process model and schedules the activities, looking for the successor(s) of a task as soon as it completes.
- In the real practice, the process model of a workflow goes through different refinements over time, e.g. due to corrective and to perfective maintenance, leading to different versions of the same process model.
- The wash-out policy (wait for completion of all running cases of the old schema before any new case can be started according to the latest schema) is not always acceptable.

Workflow Components and Temporalities

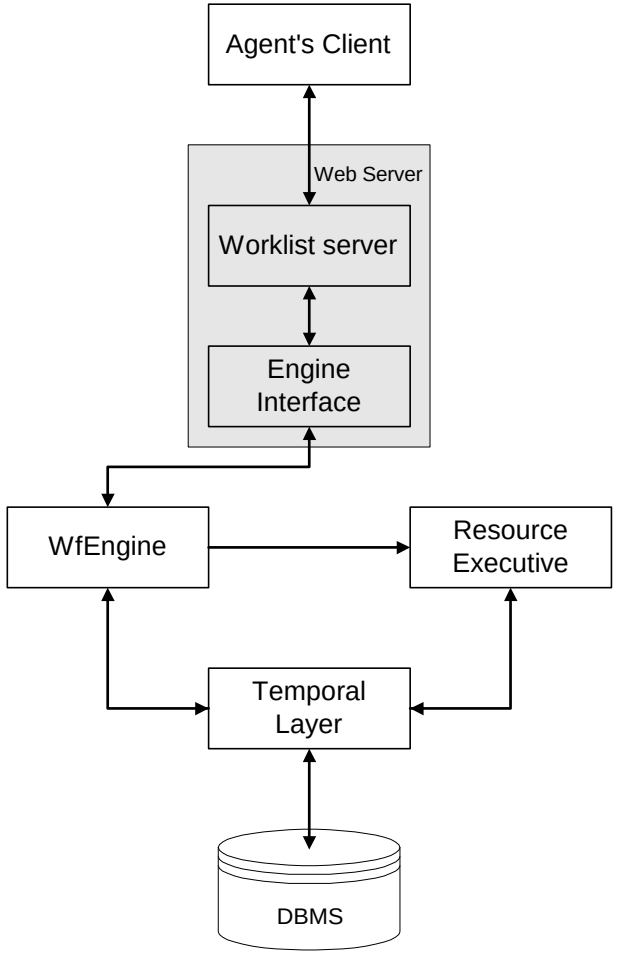
- A reasonable policy is that of completing all running cases according to the schema that was valid at their respective creation time, and to start new cases according to the latest available schema. This policy requires a temporal query which, in turn, can be easily defined if a temporal database system is available.
- Additionally the *workflow engine* manages the history of all the cases run by the WfMS, by suitably updating the CaseHistory and TaskHistory tables.
- The *resource executive* assigns tasks to agents: criteria for agent selection are statical (e.g. the role the agent must own, the agent's availability in terms of working time during weekdays), and dynamical (e.g. workload balancing among agents over the last two weeks).

Architectures for a WfMS Managing Time

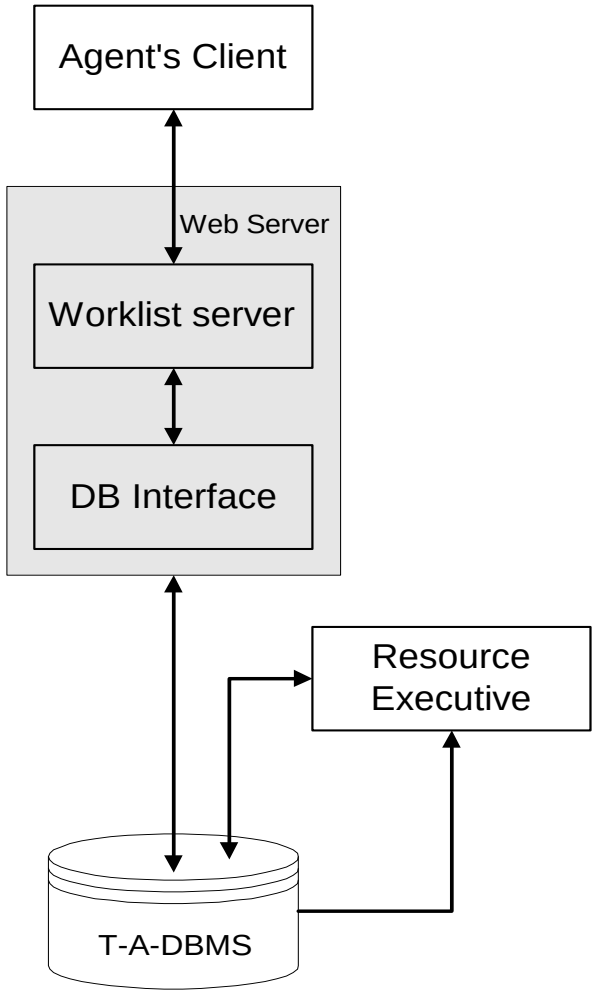
Some options:

- flat DBMSs vs fully-fledged temporal DBMSs (T-DBMSs);
- standard DBMSs vs active DBMSs.

1) A Temporal Architecture



2) A Temporal Active Architecture

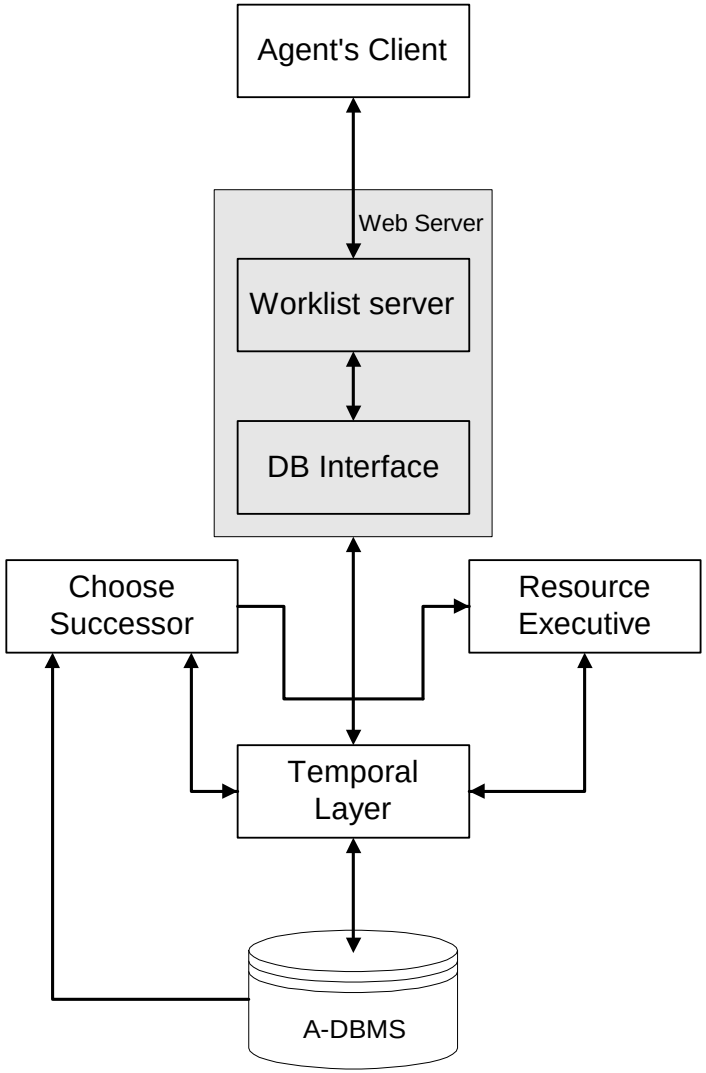


Temporal active features for workflows

Assuming that there is a straight connection between the completed task and its successor (i.e., there is no fork or join connector in between), the trigger `FindSuccessor` can be expressed in the extended Chimera-Exception language as:

```
define trigger FindSuccessor
   event       insert into TaskHistory
   conditions  TaskHistory TH, ToAssign TA,
               WorkTask WT, CaseHistory CH, Next N,
               occurred(insert(TH)),
               TH.FinalState=‘‘Completed’’,
               CH.CaseID=TH.CaseId,
               N.SchemaName=WT.SchemaName,
               N.TaskName=TH.TaskName,
               VALID(WT) CONTAINS BEGIN(VALID(CH)),
               VALID(N) CONTAINS BEGIN(VALID(CH))
   action      insert into ToAssign(CaseId,TaskName,
                   Role,AgentId)
               values TA.CaseId=TH.CaseId,
               TA.TaskName=N.NextTask,
               TA.Role=WT.Role,
               TA.AgentId=NULL,
               VALID PERIOD(current_time(), +∞)
end trigger
```

3) an Active Architecture with a Temporal Layer

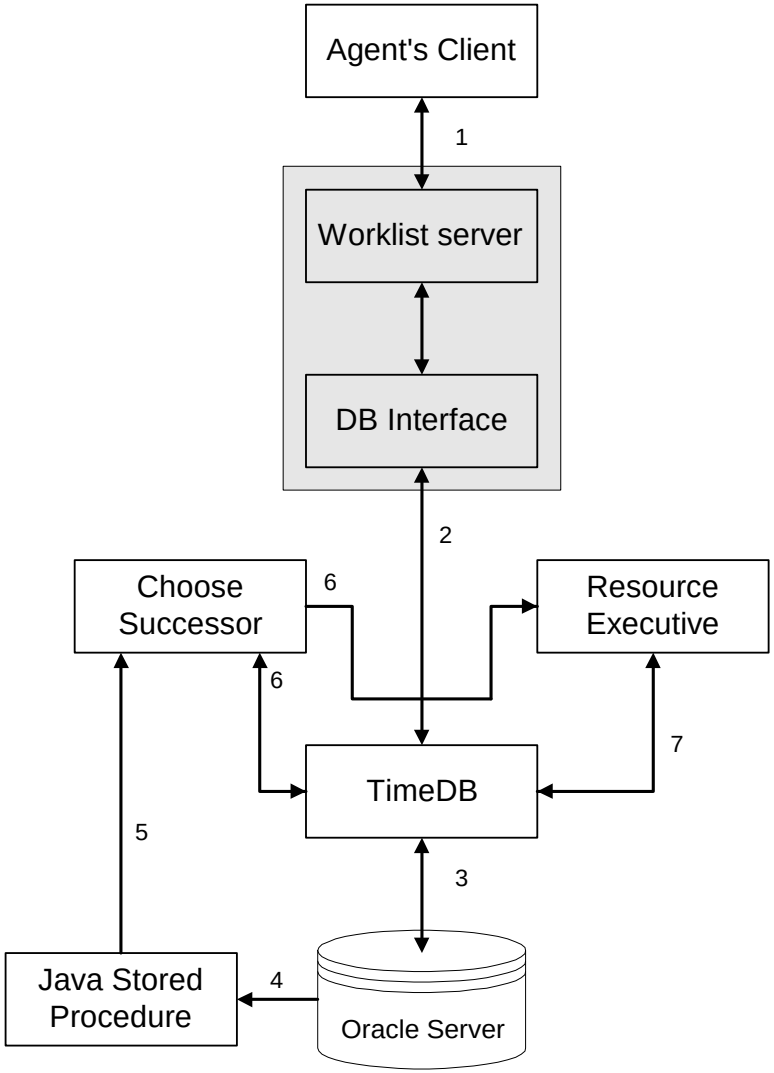


Active features for workflows

The trigger ActivateChooseSuccessor is fired when an `insert` is performed over the TaskHistory table, and that simply activates a stored procedure, namely `ChooseSuccessor`.

```
define trigger ActivateChooseSuccessor
   event       insert into TaskHistory
   condition   NULL
   action      activate(ChooseSuccessor)
end trigger
```

Implementation Issues



Implementation issues

Let us consider the functionality of choosing a successor as performed by the *workflow engine*.

- The agent sends back to the worklist server the completed task: the *DB interface* module automatically updates the workflow history tables via the TimeDB layer.
- The Oracle trigger `ActivateChooseSuccessor` has to invoke the Java stored procedure `LaunchChooseSuccessor`, which in turn invokes the `ChooseSuccessor` procedure, also written in Java, via a remote method invocation (RMI).
- The stored procedure is executed by the internal Java virtual machine of Oracle, while the `ChooseSuccessor` procedure needs for being executed outside Oracle, so that the procedure can access the database via the TimeDB layer.

Implementation issues

- In this way, the `ChooseSuccessor` procedure can perform temporal queries that could not be performed if inside the Oracle environment.
- We define a public Java class, we register it inside the database of the procedures, and define the Oracle trigger, obtaining thus the following:

```
public class LaunchChooseSuccessor{...};
create procedure LaunchChooseSuccessor
   as language Java ...;
create trigger ActivateChooseSuccessor
   after insert TaskHistory
   for each row LaunchChooseSuccessor
```

Discussions and Conclusions

- In this paper we analyzed some architectures for a temporal *workflow engine*, featuring the management of temporalities for the process, organization, and information models, and an active engine for task scheduling.
- The best architectural solution is based on the adoption of a T-A-DBMS (temporal active DBMS), providing temporal active rules. Unfortunately no available DBMS provides those features all together: we thus adopted an active DBMS on top of which we interfaced a temporal layer.
- The proposed architecture simulates temporal active rules by atemporal (timeless) active rules which, in turn, activate stored procedures and external procedures managing the temporalities via the temporal layer.

Discussions and Conclusions

- This work around, however, deepens the portability problems that feature the definition of a trigger over a DBMS:
  - (a) no SQL-like standard is available yet for the trigger definition language, and the adoption of a new DBMS requires a new set of triggers;
  - (b) very often the language used for the stored procedures is strictly DBMS-dependent, and again a change in the adopted DBMS requires rewriting the stored procedures.

Future work

- Extending the engine to adopt many different schema migration policies: *concurrent to completion* which is the only one currently implemented, *conditional or unconditional migration to final workflow*, and *migration to ad-hoc workflow*.
- Defining a temporal interpreter for active rules.
- Extending obtained results to an open source WfMS.