Temporal Workflow Management Systems

Carlo Combi[1] Giuseppe Pozzi[2]

[1]Department of Computer Science, University of Verona

[2]Department of Electronics and Information, Politecnico of Milan

Padova 2006

Outline

Outline

Business Processes and WfMSs

- 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.

Workflow systems

- Workflows are activities involving the *coordinated execution* of
  - single *tasks*;
  - performed by different *agents*.
- The *workflow specification* describes the component tasks, their controlled and coordinated execution, the processing entities.
- Workflow management systems (*WfMS*) support the automatic execution of workflows. A WfMS provides:
  - the process model;
  - the information model;
  - the organizational model.

Workflow systems

- Workflows are activities involving the *coordinated execution* of
  - single *tasks*;
  - performed by different *agents*.
- The *workflow specification* describes the component tasks, their controlled and coordinated execution, the processing entities.
- Workflow management systems (*WfMS*) support the automatic execution of workflows. A WfMS provides:
  - the process model;
  - the information model;
  - the organizational model.

Workflow systems

- Workflows are activities involving the *coordinated execution* of
  - single *tasks*;
  - performed by different *agents*.
- The *workflow specification* describes the component tasks, their controlled and coordinated execution, the processing entities.
- Workflow management systems (*WfMS*) support the automatic execution of workflows. A WfMS provides:
  - the process model;
  - the information model;
  - the organizational model.

WfMS data models and temporal information

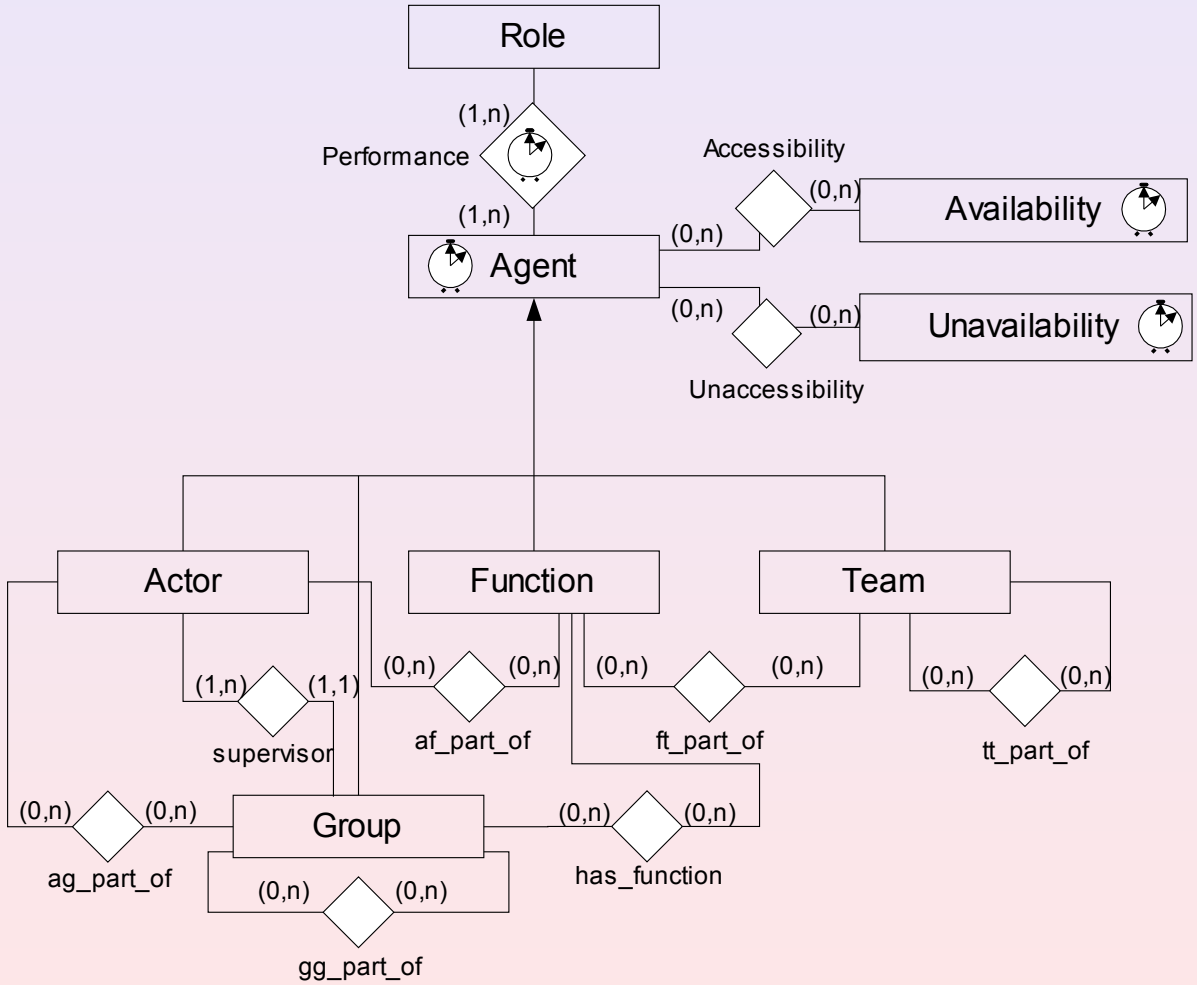
- **Organizational model**. Temporal aspects are mainly related to working days and hire time of the agent.
- **Process model**. Temporal aspects are related to changes of a schema of a business process.
- **Information model**. The information model considers both process specific and historical data. 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.

Goal of the work

- Discuss *temporal aspects for WfMS data models*
  - *Valid Time*;
  - *Temporal Constraints*.
- Discuss *different architectures for a temporal WfMS*.
- Deal with some *implementation issues*.

Outline

The temporal organizational model



The temporal organizational model

Example

Agent

Id_A	Name	Email	...	VT
1	Jane Austen	jane.austen@literature.com	...	[10/01/2000 ÷ +∞]
2	James Joyce	james.joyce@literature.com	...	[10/01/2000 ÷ +∞]
3	Emily Bronte	emily.bronte@literature.com	...	[01/03/2001 ÷ +∞]
4	Charlotte Bronte	charlotte.bronte@literature.com	...	[10/01/2001 ÷ +∞]
5	Charles Dickens	charles.dickens@literature.com	...	[10/04/2000 ÷ +∞]
6	Samuel Beckett	samuel.beckett@literature.com	...	[10/01/1999 ÷ +∞]

Role

Id_R	Description
1	Secretary
2	Committee Member
3	Committee President

Performance

Actor	Role	VT
1	1	[10/01/2000 ÷ +∞]
2	2	[10/01/2000 ÷ +∞]
3	1	[01/03/2001 ÷ +∞]
4	1	[10/01/2001 ÷ 31/08/2002]
4	2	[01/09/2002 ÷ +∞]
5	3	[10/04/2000 ÷ +∞]
6	2	[10/01/1999 ÷ +∞]

Periodicity and agent (un)availability

Availability/unavailability of an agent may present some periodicity.

Example

- ”every Monday 8:00a.m.÷4:30p.m.”;
- ”the first Wednesday of every month 1:30p.m.÷5:30 p.m.”;
- ”every day 8:30a.m.÷4:30p.m.”.

The supported periodic expressions: notations and examples

Expression	P	Example
[P]/Days_in_Weeks	x such that x ∈ [1,..,7]	[1]/Days_in_Weeks = every Monday
	1..x such that x ∈ [2,..,7]	[1..3]/Days_in_Weeks = the first 3 days of the week
	x,y such that x,y ∈ [1,..,7]	[1,3]/Days_in_Weeks = every Monday and every Wednesday
[P]/Days_in_Months	1..p such that p ∈ [2,..,M], M ∈ [28,..,31]	[1..5]/Days_in_Months = the first 5 days of the month
	p,t such that p,t ∈ [1,..,M] M ∈ [28,..,31]	[2, 6]/Days_in_Months = day 2 and day 6 of the month
	$\bar{w}$ such that $\bar{w}$ ∈ [1,..,7]	[$\bar{1}$]/Days_in_Months = the first Monday of the month
	p such that p ∈ [1,..,M], M ∈ [28,..,31]	[20]/Days_in_Months = day 20 of the month
[P]/Weeks_in_Months	1..q such that q ∈ [2,..,5]	[1..2]/Weeks_in_Months = the first 2 weeks of the month
	q such that q ∈ [1,..,5]	[1]/Weeks_in_Months = the first week of the month
	q,r such that q,r ∈ [1,..,5]	[2,4]/Weeks_in_Months = the second and the fourth week of the month

Availabilities and unavailabilities

Example

Availability

Id_Av	Start_Time	End_Time	Time_Av	VT
1	09:00	12:00	DAILY_TIME	[10/01/2000 ÷ +∞]
2	10:00	13:30	DAILY_TIME	[10/01/2000 ÷ +∞]
3	09:00	14:00	DAILY_TIME	[01/03/2001 ÷ +∞]
4	10:00	15:30	PERIODIC_TIME	[10/04/2000 ÷ +∞]
5	12:00	17:30	DAILY_TIME	[10/01/2000 ÷ +∞]
6	09:00	17:30	DAILY_TIME	[10/01/1999 ÷ +∞]

Accessibility

Agent	Availability
1	1
2	2
3	3
4	5
5	4
6	6

Availabilities and unavailabilities

Example

Unav

Id_Unav	Description	Start_Time	End_Time	Time_Unav	VT
1	Holiday	09:00	17:30	DAILY_TIME	[03/08/2001 ÷ 03/08/2001]
2	BusinessTrip	09:00	17:30	PERIODIC_TIME	[01/09/2002 ÷ 30/10/2002]
3	Holiday	09:00	17:30	DAILY_TIME	[10/08/2001 ÷ 20/08/2005]
4	BusinessTrip	09:00	17:30	PERIODIC_TIME	[01/09/2003 ÷ 30/11/2003]

Inaccessibility

Agent	Unavailability
1	1
2	4
3	3
4	3
5	2
6	3

Periodic_Time

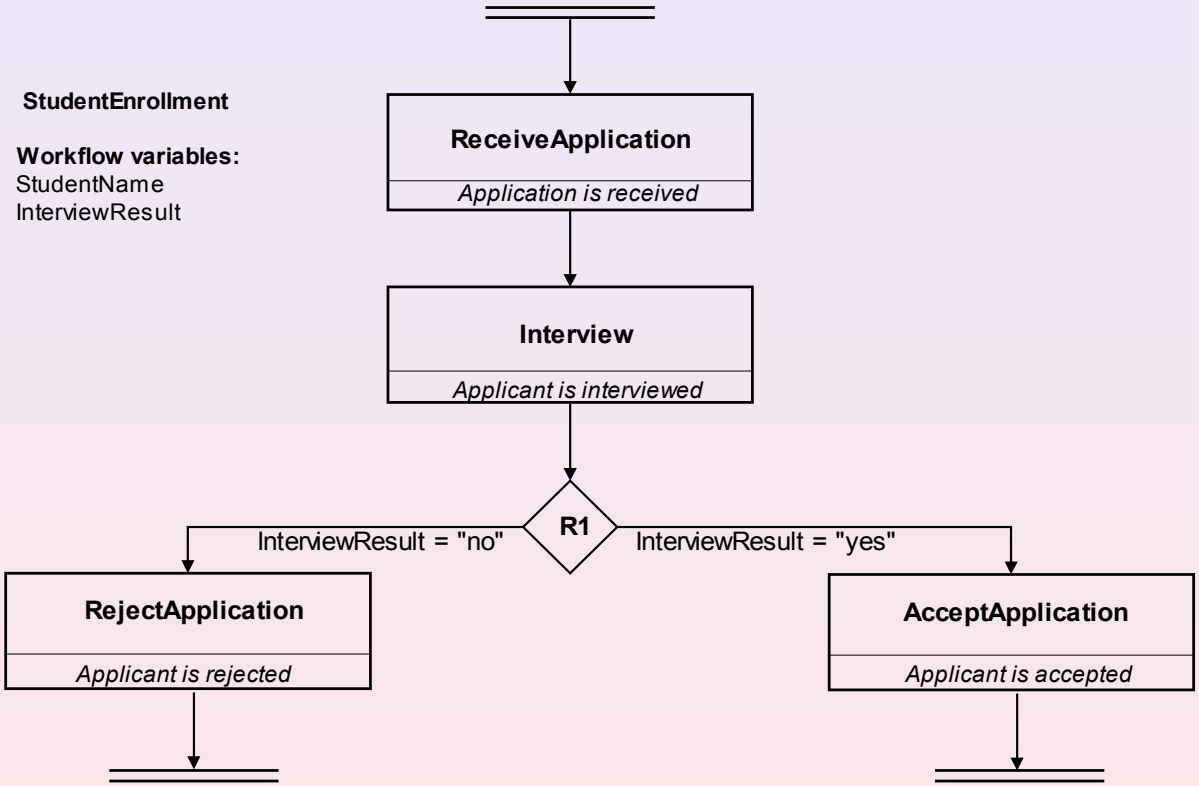
Id	Type	Expression
4	Av	[5,6]/Days_in_Weeks
2	Unav	[2]/Weeks_in_Months
4	Unav	[1..5]/Days_in_Months

The process model: an example

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.

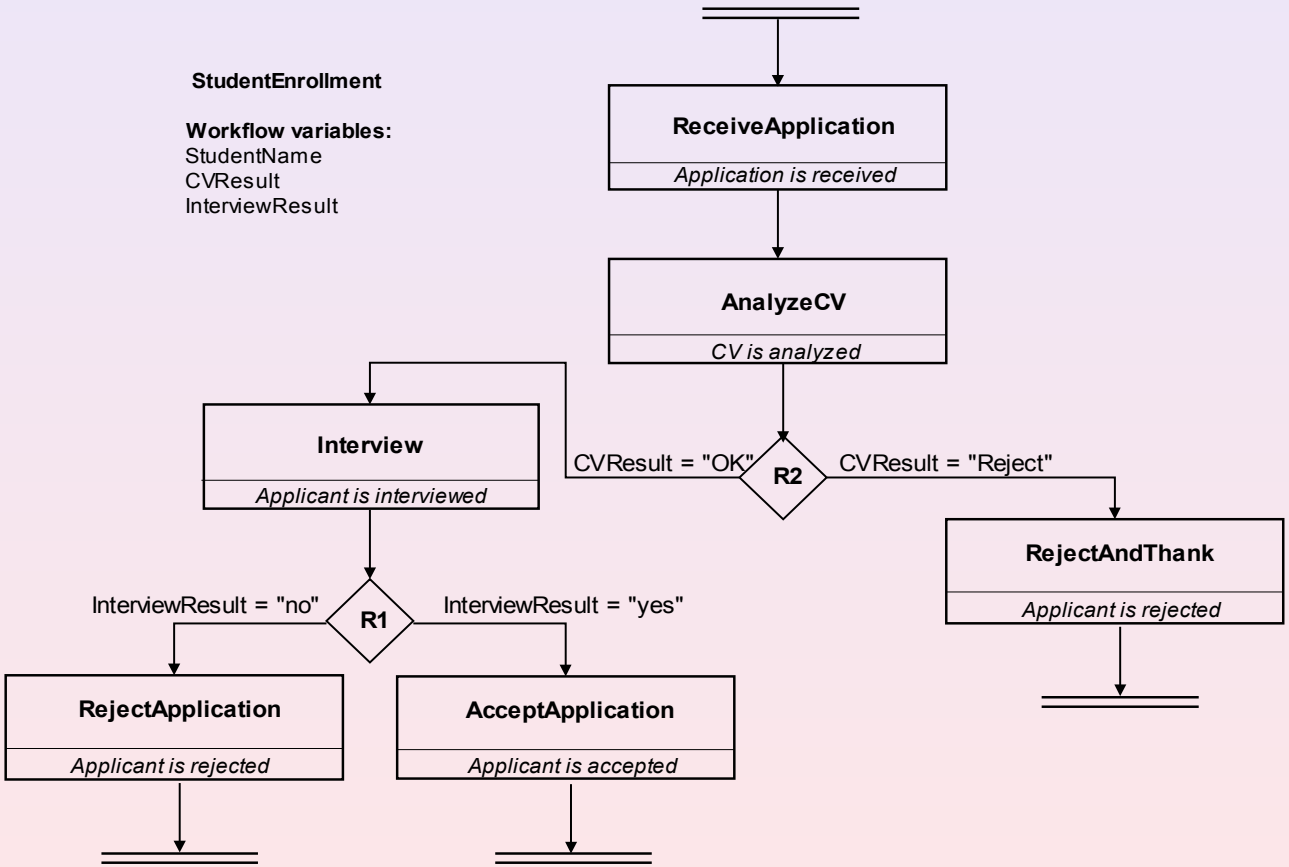
Changes in the process model

The enrollment process: first version



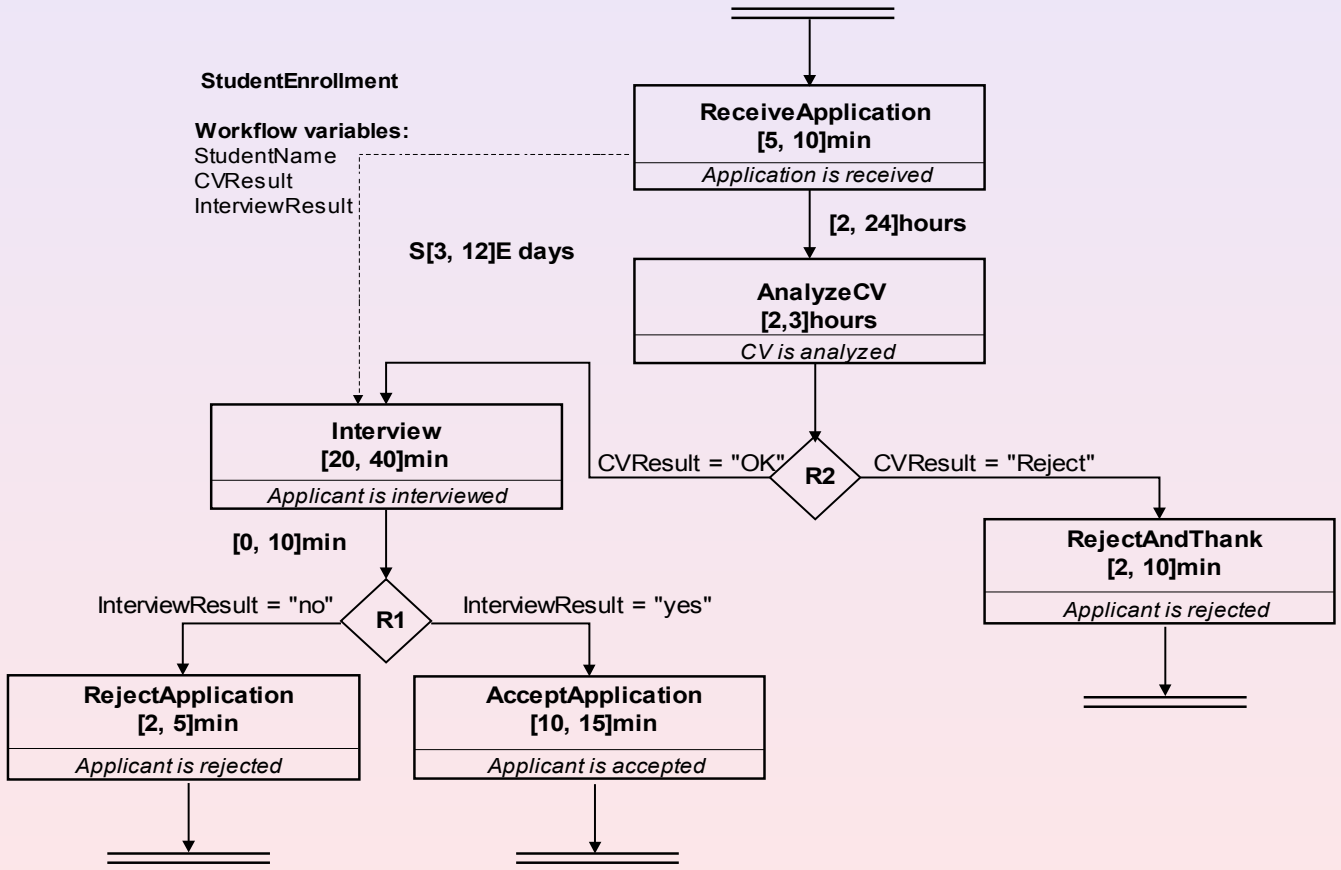
Changes in the process model

The enrollment process: second version



Temporal constraints in the process model

The enrollment process: temporal constraints



Temporal tables for the process model (1)

Example

Workflow

SchemaName	StartTask	VT
StudentEnrollment	ReceiveApplication	[01/09/2003÷ $+\infty$]

WorkTask

SchemaName	TaskName	Role	Tcons	VT
StudentEnrollment	ReceiveApplication	1	...	[01/09/2003 ÷ $+\infty$]
StudentEnrollment	Interview	2	...	[01/09/2003 ÷ $+\infty$]
StudentEnrollment	RejectApplication	3	...	[01/09/2003 ÷ $+\infty$]
StudentEnrollment	AcceptApplication	3	...	[01/09/2003 ÷ $+\infty$]
StudentEnrollment	AnalyzeCV	2	...	[01/10/2003 ÷ $+\infty$]
StudentEnrollment	RejectAndThank	3	...	[01/10/2003 ÷ $+\infty$]

Temporal tables for the process model (2)

Example

Next

SchemaName	**TaskName**	**NextTask**	**Tcons**	**VT**
StudentEnrollment	ReceiveApplication	Interview	...	[01/09/2003 ÷ 30/09/2003]
StudentEnrollment	Interview	R1	...	[01/09/2003 ÷ +∞]
StudentEnrollment	RejectApplication	end_flow	...	[01/09/2003 ÷ +∞]
StudentEnrollment	AcceptApplication	end_flow	...	[01/09/2003 ÷ +∞]
StudentEnrollment	RejectAndThank	end_flow	...	[01/10/2003 ÷ +∞]
StudentEnrollment	ReceiveApplication	AnalyzeCV	...	[01/10/2003 ÷ +∞]
StudentEnrollment	AnalyzeCV	R2	...	[01/10/2003 ÷ +∞]

Temporal tables for the process model (3)

Example

RoutingTask

SchemaName	**RTName**	**Type**	**VT**
StudentEnrollment	R1	mutualex_fork	[01/09/2003 ÷ +∞]
StudentEnrollment	R2	mutualex_fork	[01/10/2003 ÷ +∞]

AfterFork

SchemaName	**ForkTask**	**NextTask**	**Cond**	**VT**
StudentEnrollment	R1	RejectApplication	InterviewResult = “no”	[01/09/2003 ÷ +∞]
StudentEnrollment	R1	AcceptApplication	InterviewResult = “yes”	[01/09/2003 ÷ +∞]
StudentEnrollment	R2	Interview	CVResult = “OK”	[01/10/2003 ÷ +∞]
StudentEnrollment	R2	RejectAndThank	CVResult = “Reject”	[01/10/2003 ÷ +∞]

Temporal tables for the information model

Example

StudentEnrollmentData

CaseId	**StudentName**	**CV Result**	**InterviewResult**	**VT**
27	Marple T.R.	n/a	"yes"	[09/09/2003 ÷ $+\infty$]
89	Wallace E.S.	"yes"	"no"	[03/10/2003 ÷ $+\infty$]

CaseHistory

CaseId	**SchemaName**	**Resp**	**Start**	**End**	**VT**
27	StudentEnrollment	1	10:03	13:25	[09/09/2003 ÷ 27/09/2003]
89	StudentEnrollment	3	09:00	10:19	[03/10/2003 ÷ 31/10/2003]

TaskHistory

CaseId	**TaskName**	**FinalState**	**Ag**	**Start**	**End**	**VT**
27	ReceiveApplication	Completed	1	10:03	10:05	[09/09/2003 ÷ 09/09/2003]
27	Interview	Completed	4	12:06	12:25	[27/09/2003 ÷ 27/09/2003]
27	AcceptApplication	Completed	5	13:20	13:25	[27/09/2003 ÷ 27/09/2003]
89	ReceiveApplication	Completed	3	09:00	09:07	[03/10/2003 ÷ 03/10/2003]
89	AnalyzeCV	Completed	4	16:01	16:12	[10/10/2003 ÷ 10/10/2003]
89	Interview	Completed	6	09:32	10:03	[31/10/2003 ÷ 31/10/2003]
89	RejectApplication	Completed	5	10:03	10:19	[31/10/2003 ÷ 31/10/2003]

Outline

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

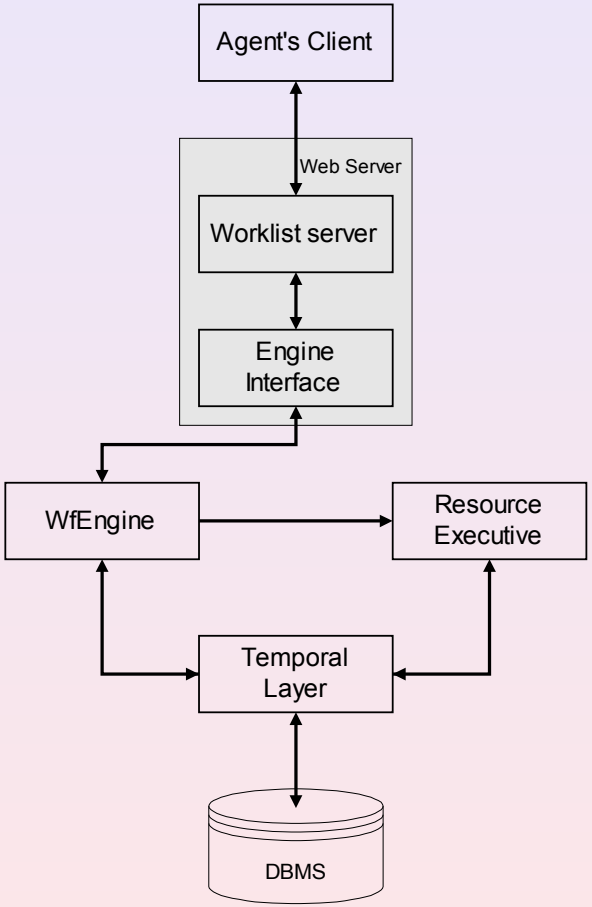
- A reasonable policy is that of completing all running cases according to the schema that was valid at their respective creation time, and to start new cases according to the latest available schema.
- Additionally the *workflow engine* manages the history of all the cases run by the WfMS, by suitably 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



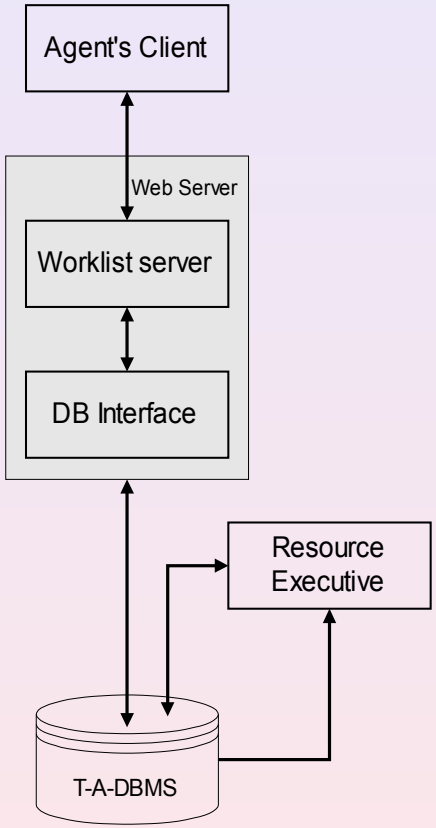
1. Temporal queries

Example

Work load balancing (over time)

```
SELECT SEQUENCED A.Id_A, T.TaskName
FROM   Agent A, Accessibility C, Availability B, TaskHistory T, Performance P1,
       Performance P2, Role R
WHERE  A.Id_A <> T.Agent AND A.Id_A = P1.Actor AND T.Agent = P2.Actor AND
       P1.Role = P2.Role AND R.Id_R = P2.Role AND C.Agent = A.Id_A AND
       B.Id_Av = C.Availability AND
       B.Start_Time <= T.Start and B.End_Time >= T.End AND
       NOT EXISTS  (SELECT SEQUENCED *
                    FROM   TaskHistory T1
                    WHERE  T1.Agent = A.Id_A)
```

2. A temporal active architecture

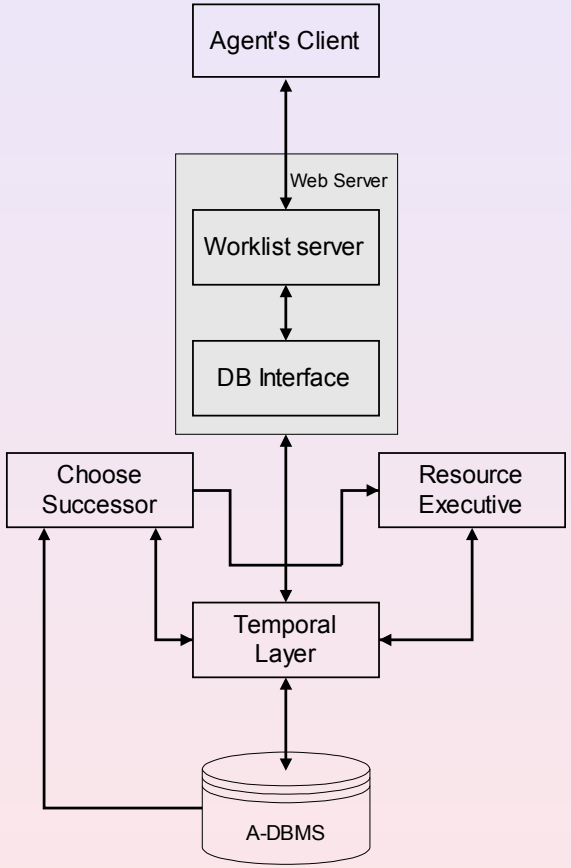


2. Temporal active features for workflows

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
   condition    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

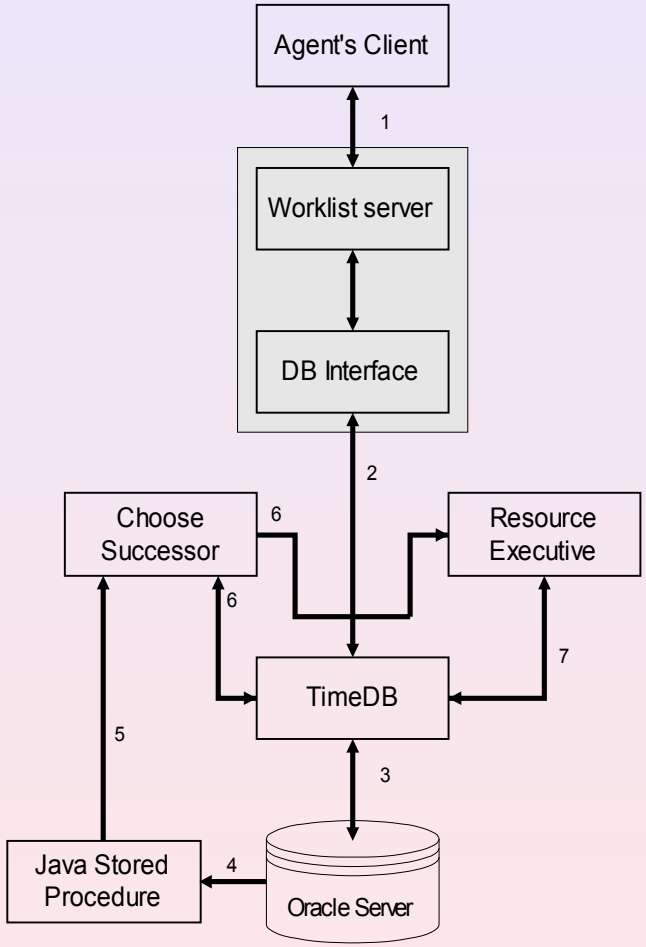


Active features for workflows

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
```

Implementation issues

- 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.

Outline

Discussions and conclusions

- In this talk we introduced and discussed temporalities for the process, organization, and information models of workflow management systems.
- We analyzed some architectures for a temporal *workflow management system*, featuring the management of the introduced temporalities.

Future work

- Verifying *run-time consistency* of cases wrt temporal constraints.
- 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.

References


C. Combi and G. Pozzi.
Towards temporal information in workflow systems.
In S. Spaccapietra, S. T. March, and Y. Kambayashi, editors, *ER (Workshops)*, volume 2503 of *LNCS*, pages 13–25. Springer, 2002.


C. Combi and G. Pozzi.
Temporal conceptual modeling of workflows.
In I.-Y. Song, S. W. Liddle, T. W. Ling, and P. Scheuermann, editors, *ER*, volume 2813 of *LNCS*, pages 59–76. Springer, 2003.


C. Combi and G. Pozzi.
Architectures for a temporal workflow management system.
In Haddad et al. *2004 ACM SAC*. ACM, 2004, pages 659–666.


C. Combi and G. Pozzi.
Task Scheduling for a Temporal Workflow Management System.
TIME 2006. IEEE Computer Society Press, pages 61-68, 2006.

References


C. Bettini, X.S. Wang, and S. Jajodia.
Temporal reasoning in workflow systems.
Distributed and Parallel Databases 11(3):269–306, 2002.


J. Eder, W. Gruber, and E. Panagos.
Temporal modeling of workflows with conditional execution paths.
In M. T. Ibrahim, J. Küng, and N. Revell, editors, *DEXA*, volume 1873 of *LNCS*, pages 243–253. Springer, 2000.

References


W. Sadiq, O. Marjanovic, and M. E. Orlowska.
Managing change and time in dynamic workflow processes.
Int. J. Cooperative Inf. Syst., 9(1-2):93–116, 2000.


P. Senkul, M. Kifer, and I. H. Toroslu.
A logical framework for scheduling workflows under resource allocation constraints.
In *VLDB*, pages 694–705, 2002.


G. B. Tramontina, J. Wainer, and C. A. Ellis.
Applying scheduling techniques to minimize the number of late jobs in workflow systems.
In Haddad et al. *2004 ACM SAC*. ACM, 2004, pages 1396–1403.