Modelling and Distributing Individual and Collective Activities in a Workflow System

Saida BOUKHEDOUMA, Zaia ALIMAZIGHI

LSI-TR 0408

Mars-2008
Modelling and Distributing Individual and Collective Activities in a Workflow System

Saïda BOUKHEDOUMA & Zaia ALIMAZIGHI

Equipe : ISI - Laboratoire des Systèmes Informatiques(LSI) - USTHB - ALGER

*: USTHB, El Alia BP n°32, Bab Ezzouar, Alger, Algérie.
Tél. / Fax : 213.21.24.79.17
sboukedouma@usthb.dz
alimazighi@wissal.dz

Abstract

This paper deals with the modelling and the dispatching of individual and collective work in a workflow system. It particularly focuses on workflow processes of organizations where actors are organized by teams. In such a system, participants must interact with each other, coordinate, exchange informations, cooperate and synchronize their work. To do that, it is essential to define firstly, the way of dispatching individual and collective parts of work to all participants in the system. We describe some static and dynamic aspects of the system and we propose a simple algorithm for managing human resources at the runtime of the process. The algorithm aims to balance the workload between human actors in the system, it is based on a workload criteria computed according to the activities durations.

Keywords


1. Introduction

With the advent and the evolution of global scale economies, organizations need to be more competitive, efficient, flexible and integrated in the value chain at different levels.
The human activity is never isolated, humans need to cooperate and communicate with each other, from that is born the concept of collective work.
The domain of collective work covers many areas of study, it has a technological dimension which aims to carry out computing devices and a human sciences dimension which focuses on sociological, psychological, economical and organizational aspects.
In the clover model, three perspectives for supporting the collective work are given: production, communication and coordination.
These three perspectives are supported by the groupware technologies and partially by workflow.
Workflow technologies have been used to support various types of business processes, allowing a partial or a global automation of them. A workflow management system covers three main functionalities which are: modelling the process, managing resources (human, material, software) and supervising the execution of process instances.
Depending on the type of workflow process, actors (participants) may work individually or collectively, in a synchronous or an asynchronous way, to perform their parts of tasks. This paper deals with the modelling and the distribution of individual and collective work of actors in a workflow system. It particularly concerns workflow processes of organizations where actors are organized by teams, and have to accomplish individual and collective activities. For example, a workflow process for planing and tracking software development, can be divided into individual and collective parts of work which are performed by individuals or teams. In this context, we establish a workflow meta-model allowing the support of concepts attached to the individual and the
collective work. At runtime of the process, actors receive activities instances materialized by work-items in their individual or their collective worklists. In such a system, participants must coordinate, exchange informations, cooperate and synchronize their work. To do that, it is essential to fairly dispatch parts of work to all actors in the system. The affectation of work-items to actors (or teams) is done by a resource manager according to an algorithm that we propose. The algorithm takes into account the workload (quantity of work) of each actor or team based on two parameters: the collective load and the individual load, in order to guarantee a load balancing in the system. The rest of this paper is structured as follows: Section 2 describes some related works dealing with the collective aspect of work, also the question of load balancing and managing resources in workflow systems. Section 3 gives some basic definitions of workflow concepts manipulated in this paper. Section 4 presents the workflow meta-model, highlighting the three aspects of the process: the organizational aspect, the informational aspect and the process aspect. In the same section, we give some dynamic views of the system, using state-transition and sequence diagrams of UML. Section 5 describes an algorithm of affecting individual and collective activities to individuals and teams. At the end of the section, a functional architecture of the system is presented. Section 6 reviews an example of a workflow process describing the planning and tracking of a software development project, and illustrates the proposed algorithm with an execution scenario on the example. Section 7 concludes the paper and gives some future works.

2. Related works

The concept of collective activity has given rise to some research in the 90’s, increasing with the emergence of communities of practices (Wenger & Snyder, 2000). In (Lorino, 2001), the author affirms that organizations are formed particularly for practical reasons, in order to implement processes that require the coordinated intervention of multiple actors. The modelling and implementing of collective activities find their origins essentially, in the educational domain. For example, (Betbeder & Tchounikine, 2003a) and (Betbeder & Tchounikine, 2003b) present a framework that aims at supporting collective activities in a learning context. (Draheim, 2003) presents a web enabled tool for the learning methodology EASE. In the domain of managing electronic documents, (Cabanac & al, 2006) propose an integrated architecture allowing a collective management of electronic documents. Many researchers such as (Soulier & Lewkowicz, 2004) and (Takamaya & al, 2007) have focused on the collaborative aspect in workflow systems. For the problem of load balancing in workflow systems, a mention may be made on (Jin & al, 2001), (Wang & Zhang, 2004) and (Katoh & al, 2004). These authors were concerned with the problem of load balancing in distributed workflow management systems, where several workflow engines must interoperate and share the execution and the control of the process activities. Other works such as (Li & al, 2003), (Li & Yang, 2005) and (Senkul & al, 2002) focused on the management of conflict access to the same resource by multiple activities. They define a model for scheduling, comprising a component for solving constraints to guarantee a proper allocation. In this work, we focus on the problem of assigning individual and collective activities to actors and teams with the aim of securing a load balancing in the system. After modelling some static and dynamic views of the system, we propose an algorithm for affecting activities to individuals and teams, based on the workload criteria.

3. Basic définitions

This section outlines the definitions of key concepts handled in this paper:

- **Workflow Process**
  A workflow process is the automation of all or part of a business process in which information flows from one activity to another (respectively, one participant to another) according to a set of predefined rules

- **Workflow Sub-process**
  This is a step in the overall process, requiring inputs and producing results (outputs). We mean that a sub-process is any internal procedure which takes part in the achievement of the overall process.

- **Individual Activity and Collective Activity**
  An activity is a step in a sub-process, requiring inputs and producing outputs. We distinguish two kinds of activities: an individual one which must be performed by one actor, and a collective one which must be performed by a team.

  It should be noted that individual and collective phases of work are done on a more or less alternated way, as shown in figure 3.1. An individual phase of work is done asynchronously and each actor must deliver his
individual result. A collective phase may possibly contain some asynchronous steps and must have inevitably, a synchronous step allowing the team members to deliver their collective result.

Figure 3.1: General schema of individual and collective phases of work

- **Resource**
  A resource is any human, material or software needed by an activity to be accomplished. In this paper we consider particularly the organization and the affectation of human resources to activities instances.

- **Actor, Team and Role**
  An actor is a human resource which participates alone or conjointly with other members of a team in the accomplishment of individual or collective activities in the process.

  A team is a set of two or more actors taking the same roles in the organization. Members of a team work collectively in order to perform collective activities, in a synchronous (and eventually an asynchronous) manner.

  A role is a qualifier assigned to an actor or a team designating its competency in the organization and so in the workflow process.

- **Resource manager**
  It is a component of the workflow system which has an overview on system resources (identifiers, roles, states, localisation, …). It is responsible for allocating adequate resources to activities initiated by the workflow engine, and releasing them at the end of activities executions. In this work, we essentially focus on human resources.

- **Worklist and Work-item**
  A worklist is a "basket" where the activities instances assigned to an actor or a team are deposited, in order to be executed. Each activity instance is materialized by a work-item. We assume that each actor accesses to its own worklist to perform individual activities. For collective activities, we assume that each team has access to a shared worklist called collective worklist.

- **Workload**
  The workload of an actor (resp. a team) corresponds to the sum of the activities durations performed or affected to the actor (resp. the team). Formulas for computing the individual workload and the collective workload are given in the sub-section 5.1.

4. Modelling Static and Dynamic aspects of the system

In this section, we describe some static and dynamic aspects of the system using UML diagrams. First, we present a meta-model for defining workflow processes, which serves as a framework for modelling processes including individual and collective parts of work.

4.1 The Workflow Meta-model

The meta-model shown in figure 4.1 highlights the main concepts of workflow processes defined by the WFMC standard such as: actor, role, process, sub-process, artefact, etc.. We add some concepts to support the collective aspect of the process like: team, collective activity, collective artefact, etc. The meta-model covers three aspects: the organizational aspect, the process aspect and the informational aspect which are detailed in the following:

---

1 Workflow Management Coalition: http://www.wfmc.org
4.1.1 The Process Aspect

This aspect describes in one hand, the cutting of the process in terms of sub-processes, activities and tasks. We see that a workflow process can be broken up into sub-processes. Each sub-process can be broken into activities and each activity can be split into tasks. The task is considered as the smallest entity of work, i.e. that cannot be broken down (see figure 4.1).

In the other hand, this aspect focuses on the intrinsic flow of control in the process, it shows the states of activities and tasks (an activity is a succession of executable tasks). An activity may be triggered by one or several events, and submitted to simple or complex conditions. We suppose that the necessary resources (tools and support documents) are available for performing activities.

The class "Activity" is specialized in two sub-classes which are "Individual activity" and "Collective activity". The first one is a part of work that must be done by one actor in an asynchronous way and which delivers an individual result. The second one is a part of work that must be done by a team in a synchronous and possibly an asynchronous way, and delivers a collective result.

4.1.2 The Informational Aspect

The informational aspect characterizes the information system part containing all data needed or produced by the execution of the process instances. Information can be pieces of databases or any other document (text, multi-media, form, file, etc.). In the current work, we just exhibit the concepts of individual and collective artefacts.

Let’s notice that an individual artefact is produced by an individual activity but can be used by an individual or a collective one. In the same way, a collective artefact is produced by a collective activity but can be used by an individual or a collective one.
4.1.3 The Organizational Aspect

This aspect represents the organizational units composing the full organization. Furthermore, it describes the affectation of resources (human actors, material and software) to these units. To support collective activities, we define the concept of team, all members of a team take the same roles in the process. A role is a concept designating the competency of actors (or teams) in the organization. Consequently, it favours a flexible distribution of activities in the system.

In figure 4.2, we present a part of the workflow meta-model at the runtime level, in order to exhibit changes affecting some classes. Some new classes like "Process Instance", "Sub-process Instance", "Activity Instance" and "Task Instance" are defined. They inherit from classes "Process", "Sub-process", "Activity" and "Task", respectively. Also, additional attributes such as "Process state", "Sub proc state", "Act state", "Actor state", "Team state" are defined to describe the changing state of dynamic objects in the system.

Furthermore, in classes "Actor" and "Team", we define two attributes "Actor_wload" and "Team_wload" in order to store the workload of actors and teams at the runtime of the process.

![Figure 4.2: Part of the workflow meta-model at Runtime](image)

During the process execution, a collective activity instance is affected to a team selected by the resource manager, and an individual one is assigned to an actor selected also by the resource manager. We suppose that the necessary material and software resources are available and assigned to the activity instance for its execution. An activity instance produces one or several artefact instances, it is submitted to a well defined condition which has a logical value.

In the following sub-section, we give some dynamic views of the system.

4.2 Dynamic Aspects of the System

In order to highlight the dynamic of an activity instance, an actor (or a team) at the runtime of the process, we describe two activities diagrams, the first one (figure 4.3) shows the life cycle of an activity instance since its selection by the workflow engine until its termination, and the second one (figure 4.4) shows the dynamic of a team (resp. an actor). We also present a sequence diagram (figure 4.5) highlighting interactions between the resource manager, the workflow engine and some objects manipulated by the system, throughout the life cycle of an individual or a collective activity instance.
At any moment, an activity instance (or a task instance) may be in one possible state "initiated", "waiting", "active", "blocked", "completed" or "resumed". We say that an activity is completed if all the tasks composing it are performed.

When an activity instance is selected by the workflow engine, it lies on the state "initiated", it goes to the state "waiting" when it is placed in the worklist of the selected actor or team (the selection is done by the resource manager), it is in the state "active" when its execution begins, until the end when it goes to the state "completed". In exceptional situations, an activity instance may pass through "blocked" and "resumed" states.

The life cycle of a sub-process is deduced from the life cycle of an activity, since a sub-process is completed if all its activities are completed. This can be generalized to a process instance.

In figure 4.4, we describe a life cycle of an actor (resp. a team).

An actor is in a "free" state if his individual worklist and the collective worklist of his team are empty, he goes to the state "sought", if a workitem is placed in his individual or collective worklist, he is in the state "busy" when he removes a workitem from its worklist and starts the execution of the corresponding activity.

In the same manner, a team is in a "free" state if its collective worklist is empty and all its members are "free". It goes to the state "sought", if a workitem is placed in its collective worklist, it is in the state "busy" when it removes a workitem from its worklist and starts the execution of the corresponding activity.

In figure 4.5, we describe all interactions between the workflow engine, the resource manager and objects manipulated by the system, throughout the life cycle of an activity instance.
Dynamic objects such as actor, team, individual activity, collective activity, individual worklist and collective worklist move from one state to another (according to transitions shown in figures 4.3 and 4.4) during the execution of an activity instance.

To affect activities instances to actors or teams, a resource manager applies an algorithm that we describe in section 5.

In the following, we propose an algorithm to be used by a resource manager for affecting individual (resp. collective) activities to actors (resp. teams).

The algorithm operates at the runtime level and aims to guarantee load balancing of work for human actors in the system. The solution is based on a property that we call $W_{load}$ designating a quantity of work (workload) assigned to an actor or a team, and computed by aggregating durations of activities affected, during a given period which may be fixed by the organization managers.

Formulas for computing the workload of an actor or a team are given in the sub-section 5.2.

5. An algorithm for affecting individual and collective activities

5.1 Notations and Formulas

$A_i$ : designates an actor identified by $A_i$

$Act_j$ : designates an activity instance

$T_k$ : designates a team number $k$.

$Roles(A_i)$ : designates the set of roles of actor $A_i$

$Roles(T_k)$ : designates the set of roles of the team $T_k$

$Roles(Act_j)$ : designates the set of roles associated to $Act_j$ (i.e roles capable of performing $Act_j$)
WL (Ai) : designates the individual worklist of the actor Ai.
WL (Tk) : designates the collective worklist of team Tk.
Associate (Actj,Ai) : indicates if the actor Ai can execute the activity Actj. It’s defined as follows:

\[
\text{if} \quad (\text{Roles}(Ai) \cap \text{Roles}(Actj) \neq \emptyset) \\
\text{Then} \quad \text{Associate} (\text{Actj}, \text{Ai}) = 1 \\
\text{Else} \quad \text{Associate} (\text{Actj}, \text{Ai}) = 0
\]

In the same manner, we define an association between activity Actj and a team Tk, as follows:

\[
\text{Associate} (\text{Actj}, \text{Tk}) : \text{indicates if a team Tk can execute the activity Actj}
\]

\[
\text{if} \quad (\text{Roles}(Tk) \cap \text{Roles}(Actj) \neq \emptyset) \\
\text{Then} \quad \text{Associate} (\text{Actj}, \text{Tk}) = 1 \\
\text{Else} \quad \text{Associate} (\text{Actj}, \text{Tk}) = 0
\]

At the runtime of the process, we define:

\[
\text{Affect} (\text{Actj}, \text{Ai}) = 1 \quad \text{if the activity instance Actj is affected to actor Ai.} \\
\text{= 0 otherwise}
\]

\[
\text{Affect} (\text{Actj}, \text{Tk}) = 1 \quad \text{if the activity instance Actj is affected to the team Tk.} \\
\text{= 0 otherwise}
\]

\[
\text{Wload(Ai)} : \text{designates the workload of an actor Ai, it is computed according to the durations of activities affected individually to the actor and those affected collectively to his team (because he takes part in performing collective activities of his team).}
\]

\[
\text{Wload} (\text{Ai}) = \sum_{j=1}^{n1} \text{duration} (\text{Actj}) \quad \text{where Affect} (\text{Actj}, \text{Ai}) = 1 \\
+ \left( \sum_{j=1}^{n2} \text{duration} (\text{Actj}) \right) /N \\
\text{where Affect} (\text{Actj}, \text{Tk}) = 1 \text{ and } \text{Ai} \in \text{Tk.} \\
N \text{ is the number of members in the team Tk.}
\]

\[
\text{Formula 5.1: Computing a workload of an actor Ai}
\]

\[
\text{Wload(Tk)} : \text{designates the workload of the team Tk, it is computed according to the durations of collective activities affected to the team Tk and those affected individually to its members.}
\]

\[
\text{Wload} (\text{Tk}) = \sum_{j=1}^{n2} \text{duration} (\text{Actj}) \quad \text{where Affect} (\text{Actj}, \text{Tk}) = 1 \\
+ \left( \sum_{j=1}^{n1} \text{duration}(\text{Actj}) \right) /N \\
\text{where Affect} (\text{Actj}, \text{Ai}) = 1 \text{ and } \text{Ai} \in \text{Tk.} \\
N \text{ is the number of members in the team Tk.}
\]

\[
\text{Formula 5.2: Computing a workload of a team Tk}
\]

5.2 Assumptions

All data manipulated by the algorithm are stored in a database which is accessible by the workflow engine and the resource manager.
For the algorithm, we put two assumptions As1 and As2:

\[
\text{As1 : Actors are organized by teams. Two actors Ai1 and Ai2 appertain to the same team if Roles (Ai1) = Roles(Ai2).}
\]
As2: It is assumed that teams are "stored" in queues, a queue Qk is constructed for all teams having the same set of roles. At any moment, Each queue is ranged in ascending order according to the teams workloads. The team at the head of the queue has a smallest (minimal) workload, in fact it is the first candidate to do the next activity associated to its roles.

5.3 The Algorithm

Begin
For each Activity instance Actj such as
   Actj.state = "initiated"
Do
Begin
If (Actj.Type = "Individual")
Begin
Construct the set E of Actors Ai / Associate (Actj, Ai) =1
   /* i.e. all actors who can execute the activity Actj */
   E = { Ai, where Associate (Actj, Ai) =1};
Construct the subset E_Wload_Min of E, containing all actors in E having a minimal workload.
   E_Wload_Min={Ai, such as Wload(Ai) = Min (wload(Ai)), Ai ∈ E}
If (|E_Wload_Min| =1) /* there is one potential actor Ai which has a minimal workload */
Then
   Affectation1 (Actj, Ai, Qk) ;
Else /* two or more actors Ai have a minimal workload */
Begin
   If exists an actor Ai ∈ E_Wload_Min And Ai.State = "free"
   Then Affectation1 (Actj, Ai, Qk) ;
   Else
   Begin
   Select an actor Ai ∈ E_Wload_Min
   Affectation1 (Actj, Ai, Qk) ;
   End ;
   End ;
End;
Else /* Actj.type = "Collective" */
Begin
Construct the set F of teams Tk / Associate (Actj, Tk) =1
   /* i.e. all teams who can execute the activity Actj */
Construct the subset F_Wload_Min of F
   F_Wload_Min = {Tk, and Wload (Tk) = Min (wload(Tk)), Tk ∈ F}
If (|F_Wload_Min| =1 ) /* there is one potential team Tk which has a minimal workload */
Then
   Affectation2 (Actj, Tk, Qk) ;
Else /* two or more teams Tk have a minimal workload */
Begin
   If exists a team Tk ∈ F_Wload_Min And Tk.State = "free"
   Then Affectation2 (Actj, Tk, Qk) ;
   Else
   Begin
   Select a team Tk ∈ F_Wload_Min
   Affectation2 (Actj, Tk, Qk) ;
   End ;
   End ;
End; End.

Page 10
/* functions Affectation1 and Affectation2 */
Affectation1 (Actj : Activity, Ai : Actor, Qk : Queue)
Begin
Add Actj in WL (Ai);
Affect (Actj, Ai) =1 ;
Wload (Ai) = Wload (Ai) + duration(Actj);
If (Ai ∈ Tk) then
Wload (Tk) = Wload (Tk) + duration(Actj)/ N; /* N is the number of the team members Tk */
Update (Qk); /* to maintain ascending order of workloads */
End;

Affectation2 (Actj : Activity, Tk : Actor, Qk : Queue)
Begin
Add Actj in WL (Tk);
Affect (Actj, Tk) =1 ;
Wload (Tk) = Wload (Tk) + duration(Actj);
For each actor Ai ∈ Tk,
Wload (Ai) = Wload (Ai) + duration(Actj)/N; /* N is the number of members in the team Tk */
Update (Qk); /* to maintain ascending order of workloads */
End;

If n is the number of activities in the process, and m is the number of human resources, the algorithm operates for each process instance in O(n*m).

5.4 Functional Architecture of the System

Figure 5.2 gives a functional architecture of the system in order to exhibit essential components like workflow engine, resource manager, worklists and the existing links between them.

Figure 5.1: Functional architecture

In the following, we give an example of workflow process including individual and collective activities, and we show a scenario illustrating the execution of the algorithm.

The example is relative to "the process of planning and tracking software development"
6 Example: A Process Of "Planing And Tracking Software Development"

For organizational aspects, we suppose that there are three roles "analyst", "designer" and "programmer", we assume the existence of 15 actors organized by teams. A team contains two or three members having the same roles in the organization, one actor appertain to one team (see table 6.1).

<table>
<thead>
<tr>
<th>Team</th>
<th>Role</th>
<th>Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Analyst</td>
<td>A1, A2, A3</td>
</tr>
<tr>
<td>T2</td>
<td>Analyst</td>
<td>A4, A5</td>
</tr>
<tr>
<td>T3</td>
<td>Designer</td>
<td>A6, A7, A8</td>
</tr>
<tr>
<td>T4</td>
<td>Designer</td>
<td>A9, A10</td>
</tr>
<tr>
<td>T5</td>
<td>Programmer</td>
<td>A11, A12</td>
</tr>
<tr>
<td>T6</td>
<td>Programmer</td>
<td>A13, A14, A15</td>
</tr>
</tbody>
</table>

Table 6.1: Teams, roles and actors

For the process aspect, we give a process model using UML activity diagram in figure 6.3 and a description of activities and tasks in table 6.2. By applying the proposed algorithm on our example for the execution of one process instance, we can obtain the scenario of execution described in table 6.3.

![Diagram of a workflow process]

Figure 6.1: An example of a workflow process

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
<th>Duration (units of tense)</th>
<th>Type</th>
<th>Implied Role</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity 1</td>
<td>Getting the knowledge of the domain and analyzing needs.</td>
<td>3</td>
<td>Coll</td>
<td>Analyst</td>
<td>Asy/Syn</td>
</tr>
</tbody>
</table>

2 A unit of tense may be a month, a week or other.
### Task 1.1
Studying the domain

### Task 1.2
Analyzing needs and writing reports

### Task 1.3
Consolidating all analyzing reports

### Activity 2
Specifying needs of the system

#### Task 2.1
Specifying needs and writing specification reports

#### Task 2.2
Preparing scenarios of acceptance tests

#### Task 2.3
Consolidating all specification reports

### Activity 3
Doing the architectural conception of the system

#### Task 3.1
Doing the global design of the system

#### Task 3.2
Establishing a global design report

#### Task 3.3
Preparing scenarios of integrating tests and integrating protocols

### Activity 4
Doing a detailed conception of Part 1

#### Task 4.1
Doing the conception of part 1

#### Task 4.2
Establishing a design report of part 1

#### Task 4.3
Establishing scenarios of unitary tests

### Activity 5
Doing a detailed conception of Part 2

#### Task 5.1
Doing the conception of part 2

#### Task 5.2
Establishing a design report of part 2

#### Task 5.3
Establishing scenarios of unitary tests

### Activity 6
Doing a detailed conception of Part 3

#### Task 6.1
Doing the conception of part 3

#### Task 6.2
Establishing a design report of part 3

#### Task 6.3
Establishing scenarios of unitary tests

### Activity 7
Programming Part 1

#### Task 7.1
Programming part 1 of the system and doing tests

#### Task 7.2
Establishing a report

### Activity 8
Programming Part 2

#### Task 8.1
Programming part 2 of the system and doing tests

#### Task 8.2
Establishing a report

### Activity 9
Programming Part 3

#### Task 9.1
Programming part 3 of the system and doing tests

#### Task 9.2
Establishing a report

### Activity 10
Integrating all parts of the system

#### Task 10.1
Integrating all parts and doing global tests

#### Task 10.2
Writing a report

---

### Table 6.2: Description of activities and tasks in the process

<table>
<thead>
<tr>
<th>Activity instance</th>
<th>Selected actor</th>
<th>Selected team</th>
<th>Workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity 1</td>
<td>T1</td>
<td></td>
<td>Wload(T1) = 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity 2</td>
<td>T2</td>
<td></td>
<td>Wload(T2) = 3</td>
</tr>
<tr>
<td>Activity 3</td>
<td>T3</td>
<td></td>
<td>Wload(T3) = 5</td>
</tr>
<tr>
<td>Activity 4</td>
<td>A9</td>
<td></td>
<td>Wload(A9) = 3</td>
</tr>
<tr>
<td>Activity 5</td>
<td>A10</td>
<td></td>
<td>Wload(A10) = 3</td>
</tr>
<tr>
<td>Activity 6</td>
<td>A6</td>
<td></td>
<td>Wload(A6) = 5/3 + 3/3 = 14/3</td>
</tr>
<tr>
<td>Activity 7</td>
<td>A11</td>
<td></td>
<td>Wload(A11) = 2</td>
</tr>
<tr>
<td>Activity 8</td>
<td>A12</td>
<td></td>
<td>Wload(A12) = 2</td>
</tr>
<tr>
<td>Activity 9</td>
<td>A13</td>
<td></td>
<td>Wload(A13) = 2</td>
</tr>
<tr>
<td>Activity 10</td>
<td>T6</td>
<td></td>
<td>Wload(T6) = 2/3 + 3 = 11/3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.3: Scenario of process execution

- Activity instance: Description of activities and tasks in the process
- Selected actor: T1, T2, T3, T6
- Selected team: Analyst, Designer
- Workloads:
  - Wload(T1) = 3
  - Wload(A1) = Wload(A2) = Wload(A3) = 3/3 = 1
  - Wload(T2) = 3
  - Wload(A4) = Wload(A5) = 3/2
  - Wload(T3) = 5
  - Wload(A9) = 3
  - Wload(T4) = 3/2
  - Wload(A10) = 3
  - Wload(T4) = 3/2 + 3/2 = 3
  - Wload(A6) = 5/3 + 3/3 = 14/3
  - Wload(T3) = 5 + 3/3 = 6
  - Wload(A11) = 2
  - Wload(T5) = 2/2 = 1
  - Wload(A12) = 2
  - Wload(T5) = 1 + 2/2 = 2
  - Wload(A13) = 2
  - Wload(T6) = 2/3
  - Wload(T6) = 2/3 + 3 = 11/3
  - Wload(A13) = 2 + 3/3 = 3
  - Wload(A14) = 3/3 = 1
  - Wload(A15) = 3/3 = 1

---

Page 13
7 Conclusion

In this paper, we were interested by modelling and dispatching individual and collective work in workflow systems. We have given some modelling aspects based on a workflow meta-model which we have extended with concepts like team, collective activity and collective artefact, to support collective aspects of work. At the organizational level, we assume that actors are organized by teams. A team is a restricted group of individuals which have the same roles in the process and so, can perform the same kind of activities. We have proposed a simple algorithm for distributing individual and collective activities to actors in order to guarantee a load balancing work between actors in the system. The algorithm is based on a workload criteria, which is computed according to the duration of activities.

As future works, we intend to implement the algorithm on a workflow system in order to check its efficiency in practice. We also aim to define an approach for scheduling activities particularly in exceptional situations such as blocking, annulation, and re-affectation.

References


