A TEMPORAL REASONING APPROACH OF COMMUNICATION BASED WORKFLOW MODELLING

J.L. Caro, A. Guevara, S. Galvez, A. Carrillo and A. Aguayo
Dpt. Lenguajes y Ciencias de la Computacion
Malaga University - 29071 Malaga (Spain)
Email: { jlcaro,guevara,galvez,carrillo,aguayo } @lcc.uma.es

Key words: workflow, workflow management systems, TLA, temporal logic, formal methods.

Abstract: Implementation of formal techniques to aid the design and implementation of workflow management systems (WfMS) is still required. We believe that formal methods can be applied in the field of properties demonstration of a workflow specification. This paper develops a formalization of the workflow paradigm based on communication (speech-act theory) by using a temporal logic, namely, the Temporal Logic of Actions (TLA). This formalization provides the basic theoretical foundation for the automated demonstration of the properties of a workflow map, its simulation, and fine-tuning by managers.

1 INTRODUCTION

The development of a workflow management system for an organization is a highly complex process. Therefore, the workflow map should be tested and validated before it is implemented; in other words, it should be analyzed prior to implementation. Most current workflow systems deal with this validation issue by using simulation modules that "execute" the model and examine the possible problems before it is truly "executed" and implemented in real life.

Although these simulation modules are very useful for the management team to detect problems in the business processes represented by the workflow, it would be advisable to find other more reliable methods. In other words, the model should allow and facilitate the automated demonstration of properties and characteristics. For example: will any workflow never be executed? Will this workflow ever be executed? Is the operation carried out with a specified time cost? Formal proving mechanisms will provide a practical solution to these kinds of problems (Hofstede et al., 1998).

In this paper we aim at approaching workflow modelling in a different way. Our object is to make a formalization of the language/action paradigm (Medina-Mora et al., 1992), based on a extension of temporal logic. This extension is known as Temporal Logic of Actions (TLA) (Lamport, 1994), and allows the easy modelling of transition states.

The application of TLA to workflow management systems provides three bases: (a) Theory: Providing a theory with a robust basis and strongly validated to carry out analyses. (b) Formalization: The possibility of formalizing workflow maps from such a theory. In other words, expressing workflow maps as TLA expressions. (c) Analysis: Providing a mechanism for the automated demonstration of workflow model properties. The capacity to prove the existence of - or freedom from - bottlenecks, deadlocks, etc.

Our approach is as follows: (i) Specification of the workflow loop semantics (section 2.2); (ii) explain the auxiliar variables used for workflow implementation, (iii) the formalization the workflow loop in TLA (section 4); (iv) formalization of basic workflow constructors.

This paper is organized as follows: we begin with a description of workflow, workflow management systems, and the modelling of workflow processes (sec. 2). In section 2.2 we analyze the basis of communication-based methodology ("speech-act"). In section 3 the TLA elements needed for the formalization are described. The core of this paper is (section 4), where the TLA formalization of the language/action paradigm is developed. The last section includes some relevant conclusions and future work.

2 WORKFLOW AND WfMS

Workflow includes a set of technological solutions aimed at automating work processes that are described in an explicit process model called the workflow map. Workflow has a wide range of possibilities as demonstrated by group support and the automation of organizational processes. In general terms we can define workflow as (Sheth and Rusinkiewicz, 1993): workflow is comprised by a set of activities dealing with the coordinated execution of multiple tasks developed by different processing entities in order to reach a common objective.

This technology is made tangible as information technology systems in the form of workflow management systems (WFMS). WFMS can be defined as (WFMC, 1994): "A system that defines, creates, and manages automatically the execution of workflow models by the use of one or more workflow engines in charge of interpreting process definitions (workflow maps), interacting with agents and, when required, invoking the use of information systems involved in the work".

2.1 Modelling techniques for workflow processes

Many authors agree on splitting workflow methodologies into two main categories (Georgakopoulos et al., 1995): (a) Activity-based methodology. These focus on modelling the activities that will take place during the development of the workflow (WFMC, 1994). (b) Communication-based methodologies. These stem from Searle's theory, known as "speech-acts" (Searle, 1975).

2.2 Communication-based methodologies

Communication-based methodologies stem from the "Conversation for Action" model developed by Medina-Mora, Winograd, and Flores (Winograd, 1988),(Medina-Mora et al., 1992). They view workflow as a sequence of conversations between a client and a server. In this section, the agents involved are described as the client requiring a service that will be developed or performed by the server .

The communication previously described between client (Cli) and server (Svr) can be defined in four steps (figure **??**): (i) Request/preparation. In this stage, the client requests an action and establishes the criteria for completing it successfully. (ii) Negotiation. In this stage, the conditions for being satisfied with the work to be done are negotiated. (iii) Development. The action is carried out by the server. (iv) Acceptance. The workflow loop is closed by accepting the work under the terms of satisfaction established in the second step.

Each stage or step can be broken down into several sub-workflows which will help to make them more specific. The set of subdivisions within the workflow loops is known as a Business Process Map (BPM).

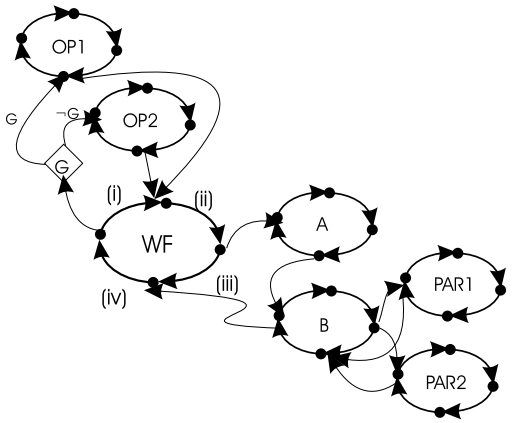

Figure 1: Example model

At any phase we can connect workflows in three modes (see figure 1): (a) Sequential mode: like A and B workflows. (b) Conditional mode: like $OP1$ and $OP2$ that can be executed depending on the guard G. (c) Parallel model: like $PAR1$ and $PAR2$.

3 TEMPORAL LOGIC OF ACTIONS

In this paper, we make use of Temporal Logic of Actions (TLA) which allows us to model state transition diagrams in a relatively easy manner. Therefore, we now describe the basic principles described by Lamport (Lamport, 1994) which are required to understand our work.

TLA combines two types of logic: Action logic, used to represent relationships between states, and temporal logic, dealing with the reasoning involved in an infinite sequence of states. All TLA formulas are TRUE or FALSE in a behavior ($\doteq$ denotes equal to by definition). We define behavior σ as an infinite sequence of states $< s_0, s_1, s_2, \cdots >$, where each state s_i has been assigned a corresponding variable.

3.1 Elements of State Logic in TLA

Variables. An infinite number of variable names (e.g., x or y) and a value class set that can be assigned to the variables are assumed. These value classes include strings, numbers, sets, and functions. If x is a

variable, $[[x]]$ is the function that semantically maps the value of x in the states. Similarly, $[[x]](s)$ is the function of the value of x in the state s.

State and predicate functions. A state function is a non-Boolean expression built from variables, constants, and standard arithmetic operators. The semantics of $[[f]]$, where f is a state function, consists of mapping states into values. To obtain the value of f in state s, we replace each variable x_i of f with $[[x_i]](s)$. Similarly, a predicate function or predicate P is a Boolean predicate. $[[P]]$ is an application of the set of states in a Boolean value. s fulfills P iff $[[P]](s)$ is equal to TRUE.

Actions. An action is a Boolean expression containing non-qualified primed variables (such as x'), standard operators, and values. An action represents an atomic operation of the system. Semantically, an action A is true or false for a pair of states, and takes the primed variables belonging to the second state. If we take an old state s, a new state t, and an action A, we obtain $[[A]](s,t)$, by first replacing each variable x with $[[x]](s)$ and each variable x' with $[[x]](t)$ to later evaluate the expression. It is said that the state pair (s,t) is a A-step iff $[[A]]$(s,t) is equal to TRUE.

Active action in a state and execution. An action A is said to be active in a state s if there is a state t such that (s,t) is a A-step (ecuacin 1).

$$[[Enabled\ A]](s) \doteq \exists t \in \sigma : [[A]](s,t) \quad (1)$$

An action A can be broken down into two logical formulae: G refers to the precondition, and B to the body of the action in itself $A \equiv G \wedge B$.

3.2 Elements of Temporal Logic in TLA

In TLA, the behavior of a system is modeled as an infinite sequence of states, where their basic elements are actions and temporal logic. In order to define the semantics of temporal formulae, we need to extend the semantic definition of the predicates whose value will be TRUE or FALSE in a given behavior. A behavior satisfies the predicate P iff (eq. 2) is satisfied in the first state.

$$[[P]](< s_0, s_1, s_2, \cdots >) \Rightarrow [[P]](s_0) \quad (2)$$

Similarly, a behavior satisfies the action A iff the first pair of states of the given behavior is an A-step (eq. 3).

$$[[A]](< s_0, s_1, s_2, \cdots >) \Rightarrow [[A]](s_0, s_1) \quad (3)$$

Always operator. The operator $\Box$ (always) is the basic block of any temporal logic. Given a formula F, $\Box F$ asserts that F is always TRUE (eq. 4):

$$[[\Box F]](< s_0, s_1, s_2, \cdots >) \doteq \forall n \geq 0 : [[F]](< s_n, s_{n+1}, s_{n+2}, \cdots >) \quad (4)$$

From equations 2 and 3 we define a behavior σ that satisfies $\Box P$ iff all the states of the behavior σ satisfy P. Similarly, a behavior σ satisfies $\Box A$ iff all steps (s_i, s_{i+1}) are A-steps.

Eventually operator. The formula $\Diamond F$ asserts that F is eventually TRUE (eq. 5):

$$[[\Diamond F]](< s_0, s_1, s_2, \cdots >) \doteq \exists n \geq 0 : [[F]](< s_n, s_{n+1}, s_{n+2}, \cdots >) \quad (5)$$

Validity. A formulae F is valid iff it is satisfied for all behaviors (eq. 6). S^∞ denotes the set of all possible behaviors.

$$\vDash F \doteq \forall \sigma \in S^\infty : [[F]](\sigma) \quad (6)$$

Specification in TLA. All previous definitions can be summed up in a single one to make a formal specification. A formal specification has the following general formula (eq. 7).

$$\Pi \doteq Init \wedge \Box(A_1 \vee A_2 \vee \cdots \vee A_n) \quad (7)$$

A formula Π is TRUE in a behavior iff its first state satisfies the predicate $Init$ and each step at least satisfies an action A_i. Actions in TLA are allowed only if the predicate $Enabled$ is TRUE and the context can be expressed as in equation 8.

$$[A]_v \doteq A \vee v' = v \quad (8)$$

This expression indicates that a new v-step is a step where either A is an A-step or the values of v do not change. Similarly, a non-stuttering execution can be defined:

$$\langle A \rangle_v \doteq A \vee v' \neq v \quad (9)$$

Fairness operators. The fairness operators are in charge of ensuring that "nothing abnormal will happen". There are two types: weak fairness (WF) and strong fairness (SF) operators.

Weak fairness. The weak fairness formula asserts that an action has to be infinitely executed frequently if it is continuously enabled for an infinitely long time.

$$WF_v(A) \doteq \Box\Diamond\langle A \rangle_v \vee \Box\Diamond\neg Enabled\langle A \rangle_v \quad (10)$$

Strong fairness. The strong fairness formula asserts that an action has to be infinitely executed frequently if it is often infinitely enabled.

$$SF_v(A) \doteq \Box\Diamond\langle A\rangle_v \vee \Diamond\Box\neg Enabled\langle A\rangle_v \quad (11)$$

Formal specification in TLA. A Φ formula represents a workflow system specification. Let be: $Init_\Phi$ variables initial state. N the execution in atomic operations form of Phi. f a used variables n–tuple. F conjunction of formulas in the form $WF_f(A)$ and/or $SF_f(A)$ where A represents an part of N. Then the specification Φ is in the following form (12).

$$\Phi \equiv Init_\Phi \wedge \Box[N]_f \wedge F \quad (12)$$

Parallel composition. If Φ and Ψ are the representation of two workflows and do not share variables the $\Phi \wedge \Psi$ represents the parallel composition.

4 FORMALIZING THE LANGUAGE/ACTION PARADIGM

As a prior step, we will define what is understood by *work*. Work w is a quadruple expressed in the equation $w \doteq \{I, H, P, SC, V\}$ where: I: Information needed to carry out the task. H: Tools and methods needed to perform the task. P: Person, role or agent with the capacity to perform the task. SC: Terms of satisfaction for the work to be considered completed. V: Set of state variables belonging to the workflow.

Each of these sub-variables is denoted as $W_k.I$, $W_k.H$, $W_k.P$, $W_k.SC$, $W_k.V$, respectively, where W_k is the work element.

Also we define a set of workflow attributes that can be used as control variables for its execution (these attributes are include in V): W_kS: State of the work. W_fS: Current state of the workflow. W_fP: Current phase of the workflow. W_k: The variable representing the current work. XW_k: External work proposed by A (where A is the workflow's client). W_fCli: Workflow client. W_fSvr: Workflow server.

At this point we can get the workflow behavior formalization. We use the example from figure 1 to illustrate how the main primitives formalize in TLA. We use the W_fP and W_fS workflow attributes for the specification. The control variable W_fP that contains the actual worflow phase is as follows:

$$W_fP \in \{\text{“}preparation\text{”}, \text{“}negotiation\text{”}, \text{“}development\text{”}, \text{“}acceptance\text{”}, \text{“}final\text{”}\}$$

W_fS represents the internal status value for a workflow. This variable is equal to “finished” when the workflows is terminated.

Also for the sub–workflows sequentiation into each phase we use the control attribute sec. This variable stores the actual execution position at each workflow phase. The sec type is an array that can be noted with $W_f.sec$ if it has only one position or $W_f.sec[1], W_f.sec[2], \ldots W_f.sec[n]$ that specifies multiples ways of execution.

4.1 Workflow loop formalization

With these elements we can formalize the case study in TLA. For the formalization we use the following notation:

- Ψ_{WF} a formal specification of workflow WF.
- WF the relationship with the following workflow state.
- $Init\Psi_{WF}$ initial condition for workflow execution. This condition is composed by $Init_{WF}$ (initial condition for work variables of workflow WF) and $Init_\Psi$ or initial condition for WF control variables. Therefore $Init\Psi_{WF} = Init_{WF} \wedge Init_\Psi$.
- $WF_{\{phase\_name\}}$ the equation that defines the behavior at workflow level for each phase. $WF_{\{negotiation\}}$ corresponds with the negotiation phase specification. If the specification does not exists its value will be $WF_{\{phase\_name\}} = \top$ and therefore always correct.
- $\Psi_{WF_{\{phase\_name\}}}$ corresponds with the phase specification.
- $Ex(WF_{\{phase\_name\}})$ corresponds with the execution level specification of $\Psi_{WF_{\{phase\_name\}}}$. This is an atomic sub–workflow and therefore equivalent at the definition level with the general formulae Ψ_{WF}. We use this term to nest specifications.
- f corresponds with the set of variables used by workflow both control variables $f_{<control>}$ and work variables $f_{<WF>}$ ($f = f_{<control>} \cup f_{<WF>}$).

To begin with the WF specification we define the general formulae Ψ_{WF} (equation 13).

$$\begin{aligned}\Psi_{WF} \equiv\ & Init\Psi_{WF} \wedge \Box[WF]_f \\ & \wedge SF(WF_{\{preparation\}}) \\ & \wedge SF(WF_{\{negotiation\}}) \\ & \wedge SF(WF_{\{development\}}) \\ & \wedge SF(WF_{\{acceptance\}})\end{aligned} \quad (13)$$

The initial state is the one at both control variables and work variables of WF (to simplify we do not

use this type of variable). This state indicates that the workflow is in the "preparation" phase (equation 14).

$$\begin{aligned} Init\Psi_{WF} &\equiv Init_{\Psi} \wedge Init_{WF} \\ Init_{\Psi} &\equiv W_f P = \text{“}preparation\text{”} \end{aligned} \quad (14)$$

The workflow execution behavior consists in a sequential execution for each phase. Each phase executes its specification; and when it finishes then the workflow transits to the next phase until the workflow has bee correctly completed (equation 15)[1].

$$\begin{aligned} WF \equiv\ & \vee WF_{\{preparation\}} \\ & \vee WF_{\{negotiation\}} \\ & \vee WF_{\{development\}} \\ & \vee WF_{\{acceptance\}} \end{aligned} \quad (15)$$

Each specification can be found in equations 16, 17, 18 and 19[2].

$$\begin{aligned} WF_{\{preparation\}} \equiv\ & W_f P = \text{“}preparation\text{”} \\ & \wedge \Psi_{WF_{\{preparation\}}} \\ & W_f P' = \text{“}negotiation\text{”} \\ & \wedge Unchg_{<f-\{W_f P\}>} \end{aligned} \quad (16)$$

$$\begin{aligned} WF_{\{negotiation\}} \equiv\ & W_f P = \text{“}negotiation\text{”} \\ & \wedge \Psi_{WF_{\{negotiation\}}} \\ & W_f P' = \text{“}development\text{”} \\ & \wedge Unchg_{<f-\{W_f P\}>} \end{aligned} \quad (17)$$

$$\begin{aligned} WF_{\{development\}} \equiv\ & W_f P = \text{“}development\text{”} \\ & \wedge \Psi_{WF_{\{development\}}} \\ & W_f P' = \text{“}acceptance\text{”} \\ & \wedge Unchg_{<f-\{W_f P\}>} \end{aligned} \quad (18)$$

$$\begin{aligned} WF_{\{acceptance\}} \equiv\ & W_f P = \text{“}acceptance\text{”} \\ & \wedge \Psi_{WF_{\{acceptance\}}} \\ & W_f P' = \text{“}final\text{”} \\ & \wedge Unchg_{<f-\{W_f P\}>} \end{aligned} \quad (19)$$

[1] We used operator $\wedge$ and $\vee$ following Lamport's recommendation (Lamport, 1994)

[2] *Unchg* is 'Unchanged operator' in TLA

4.2 Sequential tasks formalization

To continue with the example, we formalize tasks A and B where execution must be sequential. We use the sec attribute, which notes the actual execution sequence into each phase.

The general equation begins with a development phase $\Psi_{WF_{\{development\}}}$ (equation 20).

$$\begin{aligned} \Psi_{WF_{\{development\}}} \equiv\ & Init\Psi_{WF_{\{development\}}} \\ & \wedge \Box[Ex(WF_{\{development\}}]_f \\ & \wedge SF(A) \\ & \wedge SF(B) \end{aligned} \quad (20)$$

The initial state variables has an effect on the sequential step (equation 21).

$$\begin{aligned} Init\Psi_{WF_{\{development\}}} \equiv\ & WF_{\{dev.\}}.sec = \text{“}A\text{”} \\ & \wedge Init_{WF_{\{dev.\}}} \end{aligned} \quad (21)$$

The execution is based on sub-workflow specifications A and B (equations 22, 23, 24 and .

$$Ex(WF_{\{development\}} \equiv A \vee B \quad (22)$$

$$\begin{aligned} A \equiv\ & \wedge WF_{\{development\}}.sec = \text{“}A\text{”} \\ & \wedge \Psi_A \wedge WF_{\{development\}}.sec' = \text{“}B\text{”} \\ & \wedge Unchg_{<f-WF_{\{development\}}.sec>} \end{aligned} \quad (23)$$

$$\begin{aligned} B \equiv\ & \wedge WF_{\{development\}}.sec = \text{“}B\text{”} \\ & \wedge \Psi_B \wedge WF_{\{development\}}.sec' = \text{“}end\text{”} \\ & \wedge Unchg_{<f-WF_{\{development\}}.sec>} \end{aligned} \quad (24)$$

4.3 Conditional task formalization

The works $OP1$ and $OP2$ are executed depending guard G at WF preparation phase. The equation specifies how this phase works and the fairness section specifies that both $OP1$ and $OP2$ will be executed eventually if infinitely often they are active for its execution.

$$\begin{aligned} \Psi_{WF_{\{preparation\}}} \equiv\ & Init_{WF_{\{preparation\}}} \\ & \wedge \Box[Ex(WF_{\{preparation\}})]_f \\ & \wedge (WF(OP1) \\ & \vee WF(OP2)) \end{aligned} \quad (25)$$

We used G and the predicate $Init_{WF_{\{preparation\}}}$ to decide about the execution for this sub–workflows (equation 26).

$$Init_{WF_{\{prep.\}}} \equiv (G \rightarrow WF_{\{prep.\}}.sec = \text{“}OP1\text{”}) \vee(\neg G \rightarrow WF_{\{prep.\}}.sec = \text{“}OP2\text{”}) \quad (26)$$

The execution definition, using the initialitation formulae, is simple (equations 27, 28 y 29):

$$Ex(WF_{\{preparation\}}) \equiv (G \rightarrow OP1) \wedge (\neg G \rightarrow OP2) \quad (27)$$

$$OP1 \equiv WF_{\{preparation\}}.sec = \text{“}OP1\text{”} \wedge \Psi_{OP1} \; WF_{\{preparation\}}.sec' = \text{“}fin\text{”} \wedge Unchg_{<f-sec>} \quad (28)$$

$$OP2 \equiv WF_{\{preparation\}}.sec = \text{“}OP2\text{”} \wedge \Psi_{OP2} \; WF_{\{preparation\}}.sec' = \text{“}fin\text{”} \wedge Unchg_{<f-sec>} \quad (29)$$

4.4 Parallel execution formalization

The $PAR1$ and $PAR2$ workflows are executed at the execution phase of workflow B. We use the Ψ_B formulae for the specification (equation 30).

$$\Psi_B \equiv Init_{\Psi_B} \wedge \Box[B] \wedge SF(B) \quad (30)$$

The B specification follows the general schema that we used for WF (equation 31).

$$\begin{aligned} Init_{\Psi_B} &\equiv B.W_f P = \text{“}preparation\text{”} \wedge Init_{B_esp} \\ B &\equiv B_{\{preparation\}} \vee B_{\{negotiation\}} \vee B_{\{development\}} \vee B_{\{acceptance\}} \\ &\dots \\ B_{\{development\}} &\equiv \wedge B.W_f P = \text{“}development\text{”} \wedge \Psi_{B_{\{development\}}} \wedge B.W_f P' = \text{“}acceptance\text{”} \\ &\dots \end{aligned} \quad (31)$$

Finally we used the parallel composition to define $B_{\{development\}}$ (equation 32).

$$\Psi_{B_{\{development\}}} \equiv (\Psi_{PAR1} \wedge \Psi_{PAR2}) \quad (32)$$

5 CONCLUSIONS

Workflow technology needs something else other than commercial workflow products to advance process automatization. The descriptive specification of workflow maps is very useful for managers - who have a description of business processes - and for development teams. This paper introduces formal methods to carry out automated demonstrations of workflow properties. The formal method chosen is TLA. In order to prove the power of this logic, we have formalized the workflow loop of communication-based technologies. Modelling the workflow structure has been addressed at a micro-level, i.e., at the workflow loop's inner operating level.

We have shown that combining the use of traditional modelling methodologies and TLA makes possible the representation of workflow loops in a way that user-designers without expertise in logic can understand, and at the same time the model can be automatically analyzed by demonstrating workflow properties such as consistency, time deadlines, and other desirable characteristics.

REFERENCES

Georgakopoulos, D., Hornick, M. F., and Sheth, A. P. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153.

Hofstede, A., Orlowska, M., and Rajapakse, J. (1998). Verification problems in conceptual workflow specifications. *Data & Knowledge Engineering*, 24(3):239–256.

Lamport, L. (1994). The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923.

Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. (1992). The action workflow approach to workflow management technology. In *Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work*, Emerging Technologies for Cooperative Work, pages 281–288.

Searle, J. R. (1975). A taxonomy of illocutionary acts. In Gunderson, K., editor, *Language, Mind, and Knowledge. Minnesota Studies in the Philosophy of Science, Vol. 7*, pages 344–369. University of Minnesota Press, Minneapolis, Minnesota.

Sheth, A. and Rusinkiewicz, M. (1993). On transactional workflows. *IEEE Data Eng. Bull.*, 16(2):37.

WFMC (1994). Workflow reference model. Technical report, Workflow Management Coalition, Brussels.

Winograd, T. (1988). The language/action perspective. *ACM Transactions on Office Information Systems*, 6(2):83–86.