Formalizing (and Reasoning About) the Specifications of Workflows

Goce Trajcevski¹ and Chitta Baral² and Jorge Lobo³

¹ Univ. of Illinois at Chicago, Dept. of EECS, gtrajcev@eecs.uic.edu
² Bell Labs, jlobo@research.bell-labs.com
³ Arizona State Univ., Dept. of CSE, chitta@asu.edu

Abstract. We address the problem of workflow requirements specifications under realistic assumptions that it involves experts from different domains (different business policies), where not all the possible execution scenarios known beforehand. Using recent results on reasoning about actions, we formalize the notion of the specifications' correctness. To address this, we propose a high level language AWW as a basis of our prototype tool for process specification. We go “one step” before actual analysis and design, and offer a formalism which enables the experts to specify their knowledge in terms of the effects that the activities have on the workflow environment. Our methodology allows expressing not only the knowledge, but also the “ignorance” (the semantics allows unknown values to reflect a realistic situation of agents dealing with incomplete information) and the possibility of exceptional situations. We define an entailment relation which enables reasoning about the correctness of the specifications, in terms of achieving a desired goal, and testing about consequences of modifications in the workflow descriptions.

1 Introduction and Motivation

A workflow (WF) is a process which executes various cooperative and coordinated activities in order to achieve a desired goal [22]. Workflow Management Systems (WFMS) provide tools for modeling, executing and monitoring workflows [27, 40] and they need well-defined correctness/reliability criteria [1, 17, 44] and the ability to adapt to changes in a flexible manner. Recent works have addressed several of these issues [20, 25, 30, 32, 41, 50] and identified solutions of many problems of interest in WFMS. Among the other contributions, several formalisms have been proposed (e.g. OGWL (Opera Graphical Workflow Language) [30]; State and Activity Charts [50]; Concurrent Transaction Logic (CTR) [25]; Transaction Datalog (TD) [14]) which enable representation, reasoning and execution of workflows.

Since the beginning of eighties, the software engineering research has been pointing out the importance of knowledge representation being thoroughly captured during the requirements specifications stage, before moving on with analysis/design and implementation [13, 46, 47]. This is an important aspect in workflow/process specification in many applications (e.g. Virtual Enterprises) [2, 24]. The specification stage involves experts from different domains, representatives
of different organizations and policies (as Whorf's hypothesis from psycholinguistics says: "... the language a person uses to describe his or her environment is a lens through which he or she views that environment ...") [49]. Typically, not all the possible executional scenarios are known; participants need not know the details of how other partners are implementing certain (business) policies. Also, a participant may not be willing to fully reveal a decision – making process and (s)he should have the ability to choose to what extent a process will be a "black box" for the other partners (or to what extent it should be "opened"). When it comes to cooperation and synchronization, a particular domain expert typically states the "weakest preconditions" and the "strongest effects" expected.

Thus, during the early specification stage, the main problems are: has there been enough domain knowledge collected; can we reason about its correctness (which, at this stage is concerned with being able to achieve a desired goal); given the tendency of domain experts to specify mostly plausible scenarios, can we handle the occurrence of exceptional cases.

1.1 An Overview of Our Approach

In this work, we view workflows as a collection of cooperative agents and use recent results on reasoning about actions [4, 6, 29, 36] to formalize the process of their specification and test their correctness. Our approach, in a philosophical sense, resembles formalization of database updates [43] and transactions [15, 16].

Our main contributions are as follows. We present a very simple, high level language $A_W$ and, based on it, a prototype tool, which facilitates the process specification (i.e. the domain where the workflow activities will execute). The tool enables dual (textual and visual) representation of workflow specification AND enables the user to toggle between the representations at any time. The domain experts are enabled to specify the constraints on valid states and valid transitions among them, and can specify not only their knowledge but also their "ignorance" (e.g. incomplete state specification) and the possibility of abnormal (exceptional) cases. The users can express behavioral and control aspects in terms of reactive module (again, both textual and visual representations are available and an user can flip between the two at any time). With all these flexibilities (– the language allows unknown values to reflect the situations of dealing with incomplete information (essential during dynamic modifications of a workflow/Construction of ad-hoc workflows); – we give an entailment relation for querying the correctness of the specifications, and reasoning about consequences of modifications to its description ("what – if" scenarios)), $A_W$ is still based on a strict logical foundation and has a formal semantics. Hence, we have a formal notion of a correctness. of a workflow description. The rest of the paper is structured as follows. Section 2 introduces the basis of our formalism, describes the basic aspects of the language $A_W$ and our current prototype tool. In Section 3 we specify how we address the issues of cooperativity and exceptions and Section 4 presents the formalization of the notion of correctness. In Section 5 we summarize, compare with the relevant literature and propose the directions for the future work.