Available online at www.sciencedirect.com
ScienceDirect
Decision Support Systems 42 (2006) 1239–1253
Decision Support Systems
www.elsevier.com/locate/dss

On the computability of agent-based workflows

Wai Yin Mok [a,*], Prashant Palvia [b], David Paper [c]

[a] Accounting and Information Systems, University of Alabama in Huntsville, Huntsville, Alabama, United States
[b] Information Systems and Operations Management, University of North Carolina at Greensboro, Greensboro, North Carolina, United States
[c] Business Information Systems, Utah State University, Logan, Utah, United States

Received 3 January 2005; received in revised form 19 October 2005; accepted 20 October 2005
Available online 9 December 2005

Abstract

Workflow research is commonly concerned with optimization, modeling, and dependency. In this research, we however address a more fundamental issue. By modeling humans and machines as agents and making use of a theoretical computer and statecharts, we prove that many workflow problems do not have computer-based solutions. We also demonstrate a sufficient condition under which computers are able to solve these problems. We end by discussing the relationships between our research and Petri Nets, the multi-agent framework in the literature, linear programming and workflow verification.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Undecidability; Agents; Statecharts; Turing machines; Abacus programs; Workflows; Capabilities

1. Introduction

This paper investigates if computer-based decision support systems are able to solve the following research question of interest: Is a particular group of agents (humans and machines) able to achieve a business goal that is defined in terms of producing a certain number of units of a quantifiable output resource? This research question brings up a very fundamental issue. By modeling humans and machines as agents [13], we shall prove that computers cannot solve our research question, or our research question is undecidable [16], unless we restrain it in some ways. The argument of this paper is based on several assumptions. First, we define an agent framework such that no two distinct agents can execute simultaneously (see Section 3.1) and an agent can at most call on two other agents to perform work (see Section 3.2). Nevertheless, this framework has no loss of generality since it is general enough to model abacus programs, which have the same computational power as Turing machines (see Theorem 2 in Section 3.2). Second, only quantifiable resources are considered (see Assumption 1 in Section 3.2). Third, all human jobs must be simple and mundane such that they all can be completely characterized as algorithms (see Assumption 2 in Section 3.2). Fourth, time taken by a group of agents is not considered at all. Such a group may take as much time as it wants to complete its workflow (see Section 3.1). These assumptions obviously restrict this research to a narrow class of agent-based workflows. However, we shall prove in Section 3 that computers cannot even solve our research question for this narrow class of agent-based workflows. Then any broader agent framework which subsumes our framework as a special case will certainly have the same computational problems presented in Section 3 as well (see Section 4.2).

* Corresponding author.
E-mail addresses: mokw@email.uah.edu (W.Y. Mok), pcpalvia@uncg.edu (P. Palvia), david.paper@usu.edu (D. Paper).

0167-9236/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2005.10.010

The implications of this paper on workflow designs are as follows. Since business process reengineering has claimed many successes in recent years [5,6], a company might redesign its workflows for more efficiency. Since our research question is fundamental for the correctness of a workflow, any workflow designer must answer it. Nevertheless, by Theorems 3 and 4, there is no computer-based decision support system that is able to solve our research question. On the other hand, by Theorem 5 and its corollaries, if every resource required by a workflow is bounded, then computers become able to solve our research question. However, Theorem 6 shows that determining the bound for a resource required by a workflow is undecidable itself. Therefore, sooner or later human decisions must be made in a workflow design.

A word of caution is appropriate concerning this conclusion. We do not mean that computers are of no use for designing workflows. However, we do mean that designing workflows is not an exact science and many guesses have to be made during the design process. As our techniques become more accurate, most likely assisted by computers, we may be able to devise better and better workflow designs. However, our research question can never be solved *completely* no matter how sophisticated our techniques become and human decisions, which in many cases are simply guesses, must be made.

In order to prove that our research question does not have a computer-based solution, we must prove that it does not have an algorithmic solution. For this purpose, we need a formal definition of algorithms. The belief that Turing machines are sufficient to define algorithms is known as the Church–Turing Thesis. Although the Church–Turing Thesis cannot be proved because there are unlimited computational models, most mathematicians believe that the Church–Turing Thesis is true [3,16]. In this research, we make use of an equivalent computational model, namely abacus programs, to prove that some agent-based workflow problems are inherently unsolvable by computers.

Turing machines define algorithms formally. However, they are too low-level in nature to specify various features of workflows. As part of the Unified Modeling Language (UML), Harel's statecharts are used to model the reactions of a system when it faces external stimuli [2]. Moreover, statecharts have been used to model various workflow concepts specified by the Workflow Management Coalition [11]. Consequently, we chose statecharts as a vehicle to prove the assertions we make in this research. As an example, Fig. 1 shows a statechart model of a workflow in a simple garment factory. The states, which are represented as round-cornered rectangles in Fig. 1, show the basic activities needed in the workflow. The transitions, represented as arrows, are optionally guarded by conditions. Whether transitions will take place depend upon the truth and falsity of such conditions.

This paper is organized as follows. In Section 2, we establish the halting problem of statecharts and prove as corollaries some of its consequences which will be used later in the paper. Section 3 proves Theorems 3 and 4—the main results of this paper–which collectively provide a negative answer to our research question. In addition, Section 3 also proves Theorem 5 and its corollaries, which show that if every resource required by a workflow is bounded, then our research question becomes decidable. Section 4 discusses the relationships between this work and Petri Nets [12], the multi-agent framework in [15], linear programming [9] and workflow verification. The main ideas and implications are summarized in Section 5.

2. The halting problem of statecharts

An *abacus* is a theoretical computer in the sense that it has an unlimited number of registers and each register can store a number of any size [3]. No real computers

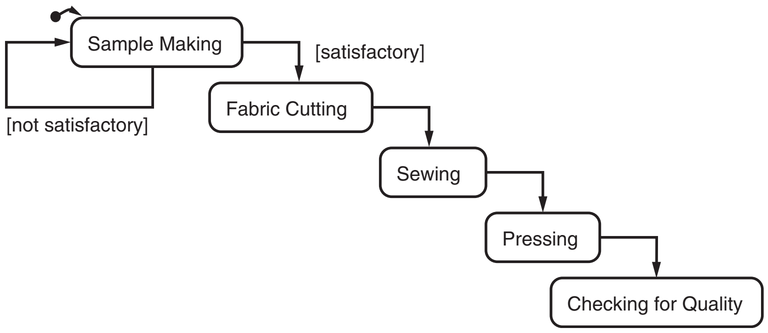

Fig. 1. A workflow in a simple garment factory.