Available online at www.sciencedirect.com

ScienceDirect

Knowledge-Based Systems 20 (2007) 152–159

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Using AI and semantic web technologies to attack process complexity in open systems

Simon Thompson *, Nick Giles, Yang Li, Hamid Gharib, Thuc Duong Nguyen

BT Research and Venturing, Adastral Park, Ipswich, United Kingdom

Received 9 October 2006; accepted 16 November 2006
Available online 11 December 2006

Abstract

Recently many vendors and groups have advocated using BPEL and WS-BPEL as a workflow language to encapsulate business logic. While encapsulating workflow and process logic in one place is a sensible architectural decision, the implementation of complex workflows suffers from the same problems that made managing and maintaining hierarchical procedural programs difficult. BPEL lacks constructs for logical modularity such as the requirements construct from the STL [STL 2003, Introduction to the STL. Available from: <http://www.sgi.com/tech/stl/stl_introduction.html>.] or the ability to adapt constructs like pure abstract classes for the same purpose. We describe a system that uses semantic web and agent concepts to implement an abstraction layer for BPEL based on the notion of Goals and service typing. AI planning was used to enable process engineers to create and validate systems that used services and goals as first class concepts and compiled processes at run time for execution.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Semantic Web; Business Processes; Agents; Planning; Abstraction; Web services; BPEL; Workflow; Tools

1. Introduction

When process flows move beyond describing the interactions required to buy a book they can become very complex. For example, the job allocation workflow implemented in BT on the CSS system in COBOL requires two sheets of A0 paper to be plotted with readable fonts. While it is possible to write a job allocation process as receive task → choose resource → allocate task real world issues such as record maintenance, audit, co-ordination, authentication, health and safety, reporting, billing, resource management, logistics and cash flow must be included to make the process operational. The move of business processes into an open environment of dynamic virtual enterprises has added another problem and another source of complexity.

Software Engineering researchers attacked the problem of complexity of process flow by developing abstractions that allowed a degree of agility and reuse to be achieved by modularity and encapsulation of data and function. Classes as an abstraction in object orientated programming provide both abstract data types and polymorphism for use cases such as the abstract factory pattern or template based development.

We have used the concepts of service mark-up and goal orientation that have been proposed by the Semantic Web and Autonomous Agent communities to implement an approach to dealing with process complexity (see Section 2) This paper describes our approach in detail with an example (Section 3) and reports on analytic tools (Section 4) and development method (Section 5) which are required to support the use of this technology in the context of developing process orientated systems. In particular our tools are designed to help engineers develop and test dynamic, open systems of services, possibly provided by Autonomous Agents.

* Corresponding author.
E-mail address: simon.2.thompson@bt.com (S. Thompson).

0950-7051/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2006.11.016

2. Previous work

BPEL is Turing Complete and is therefore an acceptable machine readable process specification. It is highly portable and widely implemented so it is an appropriate target language for our system.

The non-functional properties of the three types of future computing systems (Grid, Web Services and Ubiquitous Computing) that are currently fashionable are:

- Scale: In all three visions very large numbers of components work together over extended lifecycles.
- Openness: Components come and go dynamically, programmers have only expectations of runtime availability of functionality.
- Low cost: The hype and excitement surrounding all three visions is driven by an expectation that costs for using these infrastructures will be very low.

These non-functional properties drive the following requirements for the software development systems that are to be used to implement functionality on these three infrastructures.

- They must at least help humans find and select relevant services.
- They must be able to utilize the services that are available to achieve the human's goals.
- They must provide a high degree of automation.

Goal directed agents offer a technological response to these requirements. The canonical style of goal directed agent is the BDI agent, but it is important to stress that the canonical implementation of the BDI architecture, PRS, and PRS style agents implemented with Agentspeak and its derivatives are only one way to meet the requirements above. PRS based agents are implemented using a planner that selects scripts (plans) from a database based on the current state of the world and the current goals of the Agent. The plan database is created by the Agent developer at compile time. The Agent is therefore unable to take advantage of action effects/services that are unknown to the programmer at compile time and enter its world at run time, because by definition none of the scripts in its plan database feature these effects.

In the past many Agent systems like Decafe [7] and Zeus [5] have featured constructive planners which provide varying degrees of flexibility of behaviour in the face of dynamic sets of service actions. The problem that we have encountered in developing agents with constructive planners that the development of systems that utilize this behaviour is difficult (in the sense that PhDs find it hard), unpredictable (in the sense that committing to deadlines is difficult) and unreliable (in the sense that the system does not behave in the desired fashion). Gaia and other similar methodologies do not address agent application deployment although tools like JADE [1] and Retsina [13] do provide mechanisms to distribute and launch agents. The work that we present here integrates a methodology for agent analysis and development with a development, testing and deployment cycle.

2.1. Analytics

Process style mark-up has been used in HTN planners such as SHOP-2 [11], and in a variety of other settings, but we are unaware of any tools that provide analysis to engineers during the creation of system with ad-hoc coalition structures derived by an agent's PSM at run time. The GIPO tool is closest [10] but this is focused on planning primitives and the HTN abstraction only. The numerous algorithms that have been proposed to enable the problem solving agents operation (coalition formation for example [3], planning for example [4], coalition structure discovery for example [6]) indicate that this is a significant area of investigation, so it is surprising that so little attention has been given to designing and implementing tool support for this task.

2.2. Agent development or service composition?

Extending the discussion above we can frame the issue of dealing with the process complexity abstraction issue as that of developing technology for provisioning and utilizing (engineering with) knowledge for a situated agent in a dynamic environment. There is an extensive literature on Semantic Web service composition [9] which describes related technology.

We draw two distinctions:

1. We expect the agent utilizing the planning knowledge that it is gleaning from its domain to act *run time, and to base its compositions on the state of its beliefs about the environment* whereas service composition tools, typically, are used to generate new meta-services at compile time, which are then registered in directories as new capabilities for direct use.
2. We expect the agent to act *autonomously* in the second sense defined by Luck et al. [8]. Goal of the agent are the result of an interaction of the required outcome of a request and the context that the request is made in terms of the availability of the services and the conditions in the environment.

For example, given a scenario for a portal to provide a service for the selection of telecom services based on the features that the customer desires. New products are added, inventory changes, and the customer's circumstances change. The agent managing the portal uses its planner to provide best effort services based on the companies ability to procure and fulfil orders for the equipment and to install it in the required time windows.