# Chapter 79
# Autonomic Execution of Web Service Composition Using AI Planning Method

**Chao-Qun Yuan**
*Multimedia University, Malaysia*

**Fang-Fang Chua**
*Multimedia University, Malaysia*

## ABSTRACT

*Web Service Composition is one of the technologies in Service Oriented Architecture which significantly increases the flexibility and reusability of developing service-oriented system. One of the major problems which occurs in web service composition is the difficulties of maintaining the existing running web service composition solutions due to the changes of business requirements, deployment environment, and other dynamic factors. In this proposed work, an automated system had been built to autonomously execute the web service composition. To achieve this objective, the authors had embedded semantic engine and Prolog in C# program to automatically and dynamically discover, compose and execute web service composition, i.e. a web service composition could be self-configured to automatically recover from execution failure and automatically re-generate composition solution due to business protocol changes.*

## INTRODUCTION

Nowadays, web services have been widely deployed over the Internet as it can easily turns a software application into a web based application by publishing its functionalities and messages. The basic construction of web service platform is based on xml and http, whereby SOAP and WSDL have been used as standards for message protocol and web service description. Upon that, web service composition language such as WS4BPEL has been developed in use of composing a set of web services in designed structure to perform complex tasks (business process), whereby the web services could be executed sequentially, and parallelism and attached with pre and post conditions. However, by using web service composition languages such as WS4BPEL, it has a major drawback where the maintenance of existing composite web

DOI: 10.4018/978-1-5225-7501-6.ch079

services could be very complex and requires human intervention. Any changes on existing composition solution (caused by business requirement changes, unstable deployment environment and composition solution optimization) requires a compatibility checking on each web services in the new composition solution. These problems are commonly classified in automatic web service composition by matching the inputs and outputs of given web services, and the matching can be done either syntactically using WSDL or semantically using OWL-S. Existing works on automatic web service composition are mostly done by using AI planning methods such as situation calculus, state space planning and implemented in AI languages. By using AI Planning techniques, the activity of web service composition can be done automatically and dynamically. AI Planning represents the world model, which involves the relation among objects, actions and their causes, and reasoning about the effects of actions to find possible solutions. In this work, we proposed a complete solution including discovery, composition and execution of web service. We utilized WordNet, a lexical database for the English language (George A. Miller, 1995) to semantically map the web service operations into separate tasks and syntactically map the operations interfaces using prolog. We then represent them in an AI world model which is being implemented in Prolog using SWI-Prolog (Wielemaker, J., Schrijvers, T., Triska, M. & Lager, T. 2012). Finally, we implemented the shortest path algorithm to select the composition solution with optimized performance.

## RELATED WORKS

Several research efforts have been done on automatic service composition. (Sirin, 2010) and (Srivastava & Koehler, 2003) have proposed such method by using semantic annotated web services and AI planning to implement automatic and dynamic web service composition. (Rao & Su, 2005) and (Peer, 2005) also conducted a survey on web service composition using AI planning. In the following sections we will introduce some of the AI planning methods and their implementation in automatic web service composition.

### Situation Calculus and Golog

The situation calculus (McCarthy, 1963) is the first order language and specifically designed for representing dynamic changing worlds (Levesque, Reiter & Lesperance, 1997). The constant $S_0$ denotes the initial situation which no actions have yet occurred. A do (a,s) function denotes that an action is executed and applied in situation s. All the changes to the world are represented as actions and are defined as functions in the system. For example, putting object x on object y is represented by put(x, y), and do (put(x, y), $S_0$) function denotes that situation resulting from putting x on y when the world is in situation $S_0$. Actions in situation calculus are in first order, for example do(put(x, y), do(pickup(x), $S_0$)) denotes the situation resulting from pickup object x in situation $S_0$, and then putting object x on y. The Golog, a programming language based on situation calculus, proposed by (Levesque et al., 1997) is particularly designed for the specification and execution of complex actions in dynamic domains (Peer, 2005). Golog provides a set of extralogical (e.g., while, if, etc.) acts as abbreviations to define complex action expression in the situation calculus language. For example, in Do (δ, s, s'), δ is the complex action expression, s is the initial situation and s' is the termination situation. Sohrabi, Prokoshyna, & McIlraith (2009) have proposed a system (GologPref) that used a modified version of Golog for adapting into semantic