Paper:

Using Planning and Case-Based Reasoning for Service Composition

Chiung-Hon Leon Lee*, Alan Liu, and Huan-Hsian Huang****

*Department of Computer Science & Information Engineering, Nanhua University, Taiwan
**Department of Electrical Engineering, National Chung-Cheng University, Taiwan
E-mail: chlee@mail.nhu.edu.tw, aliu@ee.ccu.edu.tw
[Received October 14, 2009; accepted May 19, 2010]

**Planning commonly applied to automating Web Service composition involves two problems –
(i) overlooked user needs combined with services provided by the systems themselves and outside services providing a much more flexible service model.
(ii) "Speeding up" and "facilitating" services by not recording information about service providers already having served users and about planning already processed. We propose merging internal and external services to meet user needs. Internal services include system functions designed to meet user needs. External services mean Web services provided by outside service providers. We plan to combine both types of services to create planning to meet user needs. We apply case-based reasoning to store planning and related information in a case base to make planning much faster when users have similar needs.**

Keywords: service composition, planning, case based reasoning

1. Introduction

Computer systems have long proposed providing users with more varied and richer services in ways as easy as making requests from a human assistant. Service-oriented computing [1, 2] is emerging as a new and promising computer paradigm. The Web's loosely coupled reusability make it as a good choice for enhancing computer service capabilities through technologies such as SOAP, WSDL, and UDDI, for example. To fully meet business application requirements, however, current technologies still must overcome security, composition, and semantic problems [1].

One attractive Web service is composition in which simple services are found, selected, and reorganized into value-added composite services to provide users with more convenient services and solve more complex problems [3–5].

Semantic Web services [6] solve Web service problems semantically and address Web services descriptions as a whole [7]. Semantic markup languages such as OWL-S and its predecessor DAML-S [8] describe Web service capabilities and contents in a computer-interpretable language and improve service discovery, invocation, composition, monitoring, and recovery quality. An agent is a software entity having human properties such as autonomy, reasoning, learning, and knowledge-level communication [9]. To facilitate service access, agents are widely used in Web service research [1, 4, 10], enabling users to discover, interact, and compose Web services to meet user goals and intentions.

Despite the many techniques and standards proposed to solve service provision problems, a large gap remains between human users and service providers, especially for users wanting systems to serve them automatically with minimal user interaction. Assuming, for example, that a user wants to buy a book, the system must understand the user's intent regardless of whether the user has detailed information on the book. The system must, for example, find a service helping the user get details such as the ISBN, correct title, publisher, and provider. The system must also display book information to the user, interact with the user to select the book, and provide the correct order procedure. If more than one choice exists for the same book, the system sorts choices by price, for example, to provide an inexpensive one. The book is thus eventually ordered and displayed to the user.

It is naïve to expect to directly use results returned from service providers to meet a user's request, since the system may have to ask the user for more information for suitable Web services, find or select services, or compose services and process information from them, again displaying results to the user. How to integrate Web services and system capabilities and elicit user information becomes an issue in enabling intelligent Web services. How to systematically design a service-oriented system that understands the user's service request and delivers appropriate services remain open research issues in the service-oriented research domain.

In the simple bookstore service example, we show that the user's request cannot be satisfied directly if the system determines only the book in question and orders it directly. The system must interact with the user to get more book information and process book lists from the service provider. This has motivated some researchers to propose a goal-driven approach modeling service requests from users and integrating Web services, system func-

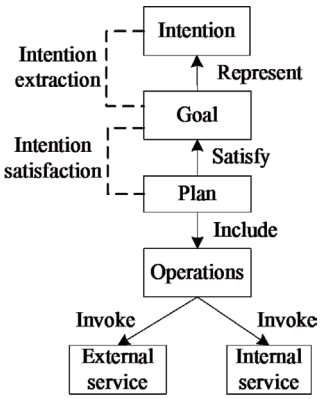

Fig. 1. Intention extraction and satisfaction.

tions, and user information to meet the request [11, 12]. How to interpret service request intent from user input is called user intention extraction and how to integrate internal and external services distributed on the Internet to meet user requests is called intention satisfaction.

We propose an approach for constructing a goal model and extracting intent from service requests [12], focusing on intention satisfaction and merging internal and external services to meet user needs. Internal service means system functions designed to meet user needs. External service means services provided by external service providers. We discuss planning [9] to combine both types of services to create planning made by a series of operations to meet user needs. We apply Case-Based Reasoning (CBR) [13] to store planning and related information in a case base to create planning much faster when users have similar needs.

The operating concepts of our proposal are shown in **Fig. 1**. Intention I is a vector consisting of a set of terms extracted from a user service request string, $I = \{T_1, T_2, \ldots, T_n\}$, in which T is terms in the service request string and n is the number of terms. Goal G abstractly describes the capability the system provides. Intention extraction problem IE is described as $IE : I \rightarrow G$, meaning that given intention I, after process IE, we map I to an abstract description of system capability G.

Plan P consists of a series of system operations. System operation OP describes a single or composite system function. Operations are used to invoke internal or external services. An internal service is a system function used to serve the user or combined with external services to generate more complex services. An external service is provided by external objects – a Web service provider or other software agents. Intention satisfaction problem IS is $IS : G \rightarrow P$, meaning that given goal G, after process IS, the system derives plan P for achieving G as follows:

Here we focus on the intention satisfaction problem, merging internal and external services to meet user needs. Internal service means system functions designed to meet user needs. External service is provided by external service providers. We apply planning to combine both services to create planning involving a series of operations to meet user needs. We apply CBR to store planning and related information in a case base to create planning much faster when users have similar needs.

This paper is organized as follows: Section 2 reviews background work. Section 3 introduces our approach to service. Sections 4 and 5 develop our proposals. Section 6 presents conclusions.

2. Related Works

Systems such as MIND [14] and Pistor [15] use planning in Artificial Intelligence (AI). Our work adds the feature of user satisfaction in service using planning and remembering the use of previous service requests, designing the use of planning in the form of Hierarchical Task Network Planning (HTNP) [14] in service composition using CBR in reusing previous requests and information.

Our prototype uses previous work on intention-aware goal model [16] and personal ontology [17] to provide services most suitable to the user. Using CBR, the system remembers how the user was previously satisfied by services to reuse previous experience in future use. HTNP is used as the base in deriving services [14]. OWL-S files provided by service providers are converted to the domain in the tool, SHOP2, and the service request entered by the user is used in planning with domain information. A description file resulting from the service in OWL-S is then produced. We use HTNP [18] for service composition, but the difference between the work above and this is that they provide an algorithm, called Enquirer, in the query manager to obtain information. The advantage is that it produces a reasonable result in initial stages when information is still lacking. An approach in planning, called model checking, is also used [19]. We use MBP Planner [20] to solve nondeterministic and partially observable problems, together with problems associated with extended goals. CBR is used [21] for service composition. We use six different relationships among services, and the service name together with its service description is used to retrieve cases to provide solutions.

Table 1 summarizes background work with features and criteria, such as main methods, capability in handling unexpected situations, capability in achieving goals, recording what service providers had used, and composition among internal and external services. All systems achieve their goals from known providers, but only some can handle unexpected situations. Since none learns providers they have previously used, all service requests are processed from scratch. No systems can compose internal and external services together. If a system has internal services, it can use such services in place of external services, saving resources in search and transmission. Such internal services may replace other external services in case of failure.

Our research uses HTN planning with CBR to provide user services. For first-time users, the system uses HTN planning to compose a service by exploring external and internal services for possible combinations. The system simultaneously learns the use through CBR. If the user is-