Realising Adaptive Web Services through Automated Policy Refinement

Kevin Carey, Vincent Wade
Dept Computer Science, Trinity College, Dublin, Ireland
Kevin.Carey@cs.tcd.ie, Vincent.Wade@cs.tcd.ie

Abstract—Traditionally policy based management systems have relied on ‘engineered’ policy refinement to decompose high level policies down to low level policies for implementation [1][2]. Such management systems typically rely on a policy designer ensuring that a high level policy creates the necessary invocations or events to trigger the correct low level policies (and their implementation). The difficulty with this approach is that human error in the policy design can lead to feature interaction problems, conflicts and errors [3], [4]. This refinement problem is a key challenge for policy based management to be adopted widely within service management (voted #1 key challenge at Policy 2005 panel session).

In this paper we concentrate on policy based management of web services, where the web services have adaptive behaviors. The paper proposes the design, implementation and evaluation of a suite of policy development tools which (i) facilitate the specification of adaptive web services (which are to be managed by a policy based system) using finite state machines to describe the permitted adaptive behaviors, (ii) supports the specification of supportable (functionally correct) high level management policies, the automated refinement of these high level policies and the auto-generation of enforceable (low level) policies. The paper also presents an evaluation of the systems with particular reference to the usability of the proposed solution. The paper also provides a brief presentation of related research in policy refinement and distinguishes the proposed approach from published work. Finally the paper concludes by drawing initial findings and identifies future work.

I. INTRODUCTION

Policy based management system are being increasingly applied across enterprise management applications such as service security, service resource allocation, storage management, and more recently, to support web service management [1][2]. However, because web services must react dynamically to changing e-business requirements, there has been a significant increase in research in the area of adaptive web services. These adaptive web services are web services which can dynamically change their behaviour based on evolving business or operating constraints. Policy based management is seen as a key way in which these potential adaptive behaviours can be managed and correctly enforced.

In this paper we focus on supporting adaptive web services, where the adaptive behaviours are intended to address changes in the non-functional requirements of the web services. Our approach allows managers to produce functionally correct enforceable policies to manage the adaptivity of these web services without needing intricate knowledge of the adaptive behaviours of the web services involved. This is achieved by specifying high level management policies and automatically refining them into enforceable (low level) policies based on the description of the adaptive behaviour(s) belonging to the web services involved. We define policy refinement as the automated process of mapping high level policy to a set of low level policies which are interpretable by the managed system, service or device. Our approach not only provides the refinement process, but also auto generates the necessary enforceable low level policies which can then be interpreted and executed on the managed system at runtime.

II. TECHNICAL APPROACH

The Web Service Definition Language (WSDL) is a standard service description language, which describes the web services interface in three parts: definitions, operations, and service bindings. However, a richer semantic language is needed in order to reason about services where such services are aggregated or composed to form a new composite service. OWL-Based Web Service Ontology (OWL-S) [9] uses ontology based semantics to enhance web service descriptions. It aims to provide an unambiguous, computer-interpretable form, rich definitions of inputs and outputs parameters with respect to an ontology, as well as a rich set of control constructs for connecting constituent services. Thus, OWL-S is intended to allow the description of web services composed of other services, including other composite services.

However, although OWL-S and WSDL can describe a web service, such specification do not readily capture the adaptive behaviours of an adaptive web service - i.e., behaviours that are not necessarily differentiated through their definition of inputs and outputs. Although OWL could be used to describe the adaptive behaviours of a service, these descriptions would lack a formalized structure. Finite State Machines model provides a formalized structure which includes states, events, and transitions. This paper proposes that a FSM model should be used by the service developers to describe the service’s adaptive behaviours which are intended to be exposed and managed via policies at run-time.

To describe an atomic adaptive web services, a generic (request-respond atomic service) FSM model is used which is composed of four composite-states – Idle State, Input State, Process State and Output State. These composite-states represent the service’s running states and were inspired by the web service description from [7]. These composite-states can
contain several FSM which represent adaptive behaviours that can be performed in a service. Thus, these FSM are categorized under a relevant service’s running state. The adaptive behaviours are described according to the FSM model described by [10].

It is important for correctness that the adaptive behaviours of a composite service do not need to be manually described by developers. Rather, the developers need only model the adaptive behaviours of the constituent atomic services. The resultant aggregation of these adaptive behaviours is based on the service composition thus assuring that the adaptive behaviours of a composite service be the adaptive behaviours of its constituent services. The FSM of the resultant composite service has the same structure as the FSMs of the atomic services.

Once the adaptive behaviours of an (atomic) web service have been modeled correctly using FSM, we propose the use obligation policies to manage these adaptive behaviours. The expectation is that administrators would be allowed to specify high-level policies which manage the adaptive behaviours of composite services. However, to ensure functionally correctness, the specification of these management policies is limited to the expressiveness of the adaptive behaviours and the web service semantic descriptions. When specifying such high level policies, the policy’s condition is expressed in terms of the service’s parameters (inputs and outputs), and the policy’s action identifies the desired service’s adaptive behaviour.

Furthermore, administrators don’t have to create low-level policies or map the specified high-level management policies to low-level policies. Instead, high level management policies are automatically refined and enforceable low-level policies are generated. The refinement of management policies is based on service’s composition, service’s description, and the FSM modeling the service’s adaptive behaviours. The approach taken generates enforceable policies for the relevant constituent service which enforces the desired adaptive behaviour. In order to allow the mapping of the high level policy to these newly generated low level policies, the system also generates mapping policies which map the specified high level management policy for a composite service to the low level policies enforceable on the relevant constituent services.

The policy refinement is accomplished in a hierarchical manner starting with the high-level policy specified for the topmost web service. Policy refinement process is performed in two parts: the refinement of the management policy’s condition aspect, and the refinement of the management policy’s action aspect. When refining the action aspect, enforceable policies are generated for the relevant constituent atomic service according to the FSM modeling the chosen adaptive behaviour. The mapping policies are deduced from the service’s composition and adaptive behaviours’ aggregation path.

When refining the condition aspect, the mapping policies are deduced from the service’s dataflow description. Enforceable policies are generated for the relevant constituent atomic service. Abstract OWL-S parameters used in the policy’s conditions are translated to WSDL parameters based on the OWL-S grounding description for the enforceable policies. These low-level policies can then be deployed together with their services so as to manage their adaptivity dynamically according to the desired intention, e.g. customer’s preference or part of a business model.

III. TOOLS

A suite of policy development tools was designed and implemented for (i) describing web services semantically, (ii) describing their adaptive behaviours, (iii) specifying management policies and automatically refining to low level policies. These tools can be used to describe adaptive web services, and to specify and automatically refine management policies into enforceable policies which manage the adaptivity of these web services. A detailed description of the innovative tools in this set of integrated tools is provided below.

A. SABE Tool

A Service Adaptive Behaviour Editor (SABE) tool was designed and implemented to allow service developers describe the adaptive behaviours of web services using FSM. This tool is not intended for describing the entire internal logic of a service, but rather to describe the adaptive behaviours of a service which the developer wishes to expose; E.g., adaptive behaviours that can accommodate user preferences, or business models. This tool uses a service’s semantic description (OWL-S) to reason about a service and to further enrich it with a description of its adaptivity using FSM. When describing the adaptive behaviours of a composite service, the SABE tool automatically aggregates the adaptive behaviours belonging to the constituent services.

This tool uses two ontology models: a web service ontology model to reason about the web service’s semantic description; a finite state machine ontology model to reason about the adaptive behaviours of a service. The service ontology (OWL-S) model was created by [9], whereas the finite state machine ontology model was derived from the UML finite state machine model [10] by the authors. This tool uses a single OWL reasoner - Jena [11], to reason about the relevant ontology instances saved as owl files.

The SABE tool was designed to allow users to easily describe the adaptive behaviours of web services as a FSM. It also auto-generates the initial FSM model with the composite-states for capturing the descriptions of a web service’s adaptive behaviours. This tool displays a FSM with the use of a tree structure together with a table to show the details of each node. When describing an adaptive behaviour, the SABE tool helps the users with dropdown menus with correct options.

The adaptive behaviours of a composite service should consist of the resultant aggregation of adaptive behaviours belonging to its constituent services. This aggregation process can be tedious, especially if a composite service is composed of services with several adaptive behaviours. The SABE tool alleviates this problem by automatically aggregating the FSMs representing the adaptive behaviours of the constituent services. The aggregation is done according to the composition description of the composite service.
B. SMPE Tool

A Service Management Policy Editor (SMPE) tool (shown in figure 1) was designed and implemented to allow high level management policies to be specified. These policies are specified to manage the adaptive behaviours, described as FSM, of a (composite) web service. Once a high level management policy is specified, this tool can then automatically refine it, auto-generating enforceable policies for the relevant constituent services.

Figure 1. SMPE tool

The SMPE tool was also designed make the specification of high level management policies as easy as possible. The left panel of this tool displays the web service’s details, the middle panel displays the adaptive behaviours of a selected web service, and the right panel displays the management policies. This tool uses a tree structure and a table to allow users to view and edit management policies.

The SMPE tool also helps the user when specifying different parts of a management policy by providing a list of possible options available, e.g. a list of available FSMS for the action aspect of a policy. Management policies used by this tool are obligation type policies with events, conditions, and actions. Boolean operators AND and OR are used to group two policy’s items together, e.g. a policy with two actions is expressed as a policy with a complex action which contains two simple policies with an AND Boolean operator.

Once the user is satisfied with the high level policy that he/she has specified, the SMPE tool performs the necessary policy refinement analysis. It analyses the high level policy and the relevant description information of the service and its adaptive behaviours. This tool is thus able to auto-generate mapping policies which maps high-level policies (for a composite service) to low-level policies based on that composite service’s OWL specification. It refines a policy by auto-generating mapping policies for the condition aspects of a policy according to the service’s dataflow, and the action aspects of a policy according to the service’s control flow. These mapping policies are event driven policies which eventually triggers the enforceable policy assigned to the relevant constituent atomic service. Currently, it only handles the refinement of policies assigned to a composite service of a linear composition, e.g. a sequence. Refinement of policies for composite services of other type of composition has not yet been verified e.g. a web service whose composition involves the parallel execution of two or more constituent web services.

Management policy’s action identifies the adaptive behaviour to be used. This policy is first refined to low level policies assigned to the relevant atomic service which contains the chosen adaptive behaviour. The auto-generated policy assigned to an atomic service is refined by this tool to a set of enforceable policies which are auto-generated based on FSM modeling the chosen adaptive behaviour. Enforceable policies are created for every transitions provided by the FSM. It, utilizes its events, guards, and action from the target state.

Parameters of a service are used in the condition aspect of a high level management policy. The difficulty is that these parameters are defined in OWL-S and as such are always abstract. However, OWL-S also provides a grounding description, i.e. a mapping between service’s semantic description and the functional description of a service in WSDL. Thus, the SMPE tool uses this information when generating the enforceable policies assigned to an atomic service which refines the condition aspect of a high level policy.

IV. Evaluation

As part of the development of the SABE and SMPE tools, usability evaluation studies were conducted. These were performed during the incremental development of the tools. Based on the usability evaluation process developed by Jacob Neilson, a small user group evaluation was conducted in each iterative development cycle of the tools. This section outlines the results of the evaluation on the SABE and SMPE tools. This evaluation study involved six postgraduate researchers acting as web service administrators/managers responsible for carrying out specific management tasks (e.g. defining adaptive behaviours of web services, composing high level policies, etc.).

The goal of this usability evaluation was to investigate how easy and understandable these tools were in capturing the adaptive behaviours of web services and in creating policies to manage them. The usability evaluation was comprised of six experiments; three experiments for testing the usability of the SABE tool to model adaptive behaviours; and three experiments for testing the usability of the SMPE tool to specify and refine management policies. A questionnaire was provided at the end of each experiment where the users answered questions about the usability experience of the tool used. The questionnaire is broken down into questions tailored to find out the user’s opinion of the different aspects of a tool after performing the tasks asked in the domain of easiness, understanding, and helpfulness.

Due to the length restrictions for this paper, we only present some key findings below,

- The task of describing adaptive behaviours as FSM, specifying management policies and refining them was
found to be easy to accomplish using the integrated tools SABE and SMPE
- FSM models were found to be well suited to describe adaptive behaviours.
- Policy was found to be suited to manage the adaptive behaviours, described as FSM, of an adaptive web service. It demonstrated how users could create functionally correct enforceable policies without in-depth of the web service’s adaptive behaviours.

V. RELATED RESEARCH

Automated Policy refinement has been the focus of several research projects and initiatives. This section illustrates some principal approaches whilst differentiating them from our proposal. One type of approach is to use machine learning techniques to support policy refinement. Verma [8] proposes a case-based reasoning approach to support policy refinement. In this approach the system learns experimentally from the operational behaviour it has previously examined. The authors have developed techniques to improve the refinement performance using clustering the different goals. This approach requires large volume of data to be accumulated of the operations performed. It differs from our approach as our tools seek to explicitly model only the adaptive behaviours and support empower the administrators/managers in composing appropriate high level management policies.

Another type of approach, suggested by Russo [5], uses a goal based approach to (semi) automate policy refinement [6]. This uses additive reasoning to derive a set of goals which satisfies the specified policy goal, creating a strategy for the policy refinement. This policy refinement approach only provides a partial automated policy refinement, since this process is automated only for the goals previously refined. This is realized by manually refining a goal using Event Calculus techniques and caching the strategy taken for this policy refinement with KAOS for future refinements. Thus, low-level policies are not auto-generated but mapped to by this approach.

VI. CONCLUSIONS

This paper has show how adaptive web services have their adaptive behaviours easily expressed as a FSM and how management policy specification and policy refinement can be used produce enforceable policies to manage these adaptive web services.

In this paper a technical approach together with a description of the tools implemented for describing adaptive (composite) web services and for specifying and refining management policies to manage these adaptive web services were presented. It also presented the results of an usability evaluation performed on the tools implemented to support the methodology proposed.

FSMs can model complex behaviours but in this case such descriptions are overpowered since they are not been used to model the entire behaviour or internal logic of a service but only to model the alternative behaviours that the developer wishes to expose; behaviours that doesn’t change the functionality of the service how it is achieved.

Policy was found to be fitting for managing the adaptive behaviours of a web service. By limiting the specification of the management policies to the web service description and the web service’s adaptive behaviours, these management policies could be automatically refined to produce low-level enforceable auto-generated policies. Since web services are atomic in nature, and its adaptive behaviours are internal, there is no necessity for the use of a confliction resolution engine to be used on the refined policies at the functional level.

This paper has shown how management policies can be specified correctly at the functional level by administrators without having in-depth expertise concerning the adaptivity of web services been managed. It showed how the proposed approach can create functionally correct enforceable policies to manage adaptive web services by automatically refining these management policies based on the web service composition and its adaptive behaviour’s descriptions. Where these descriptions are provided by experts and thus assuring the policy’s functionally correctness. The approach described in this paper could prove beneficial in allowing composite web services to be adaptive to customer’s preferences or to business needs without having to change its composition or to add more complexity to the composition description.

REFERENCES