Designing Self-Adaptive Service-Oriented Applications

Giovanni Denaro, Mauro Pezzè, Davide Tosi
Universitá degli Studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8
I-20126, Milano - Italy
{denaro|pezze|tosi}@disco.unimib.it

1 The Service Mismatch Problem

The integration of third-party web services can solve complex business problems and reduce risks, costs and time-to-market. However, the task of the integrators is challenged by services that are maintained by different organizations, and may evolve dynamically and autonomously. The impossibility of statically determining which service implementation is bound at runtime may lead to unexpected failures [2].

Consider for example, the request for a shopping cart web service, such as the one provided by Amazon [www.amazon.com]. The WSLD interface of the Amazon service matches several services that are offered by other providers, and differ in several small but important details. For instance, some services offer `CartAdd` operations that manage multiple instances of the same item in a single add operation, while others do not support multiple instances; Some services provide `CartModify` operations that remove items by resetting the available quantity to zero, while others remove items by suitably decreasing the item quantity; Some services create empty carts (`CartCreate` operation), while others require at least an item to create a cart; Some services manage orders and purchases across different user sessions (*persistent cart*), while others loose their contents when sessions terminate (*non-persistent cart*).

2 A Self-Adaptive Approach

High availability requirements and dynamic discovery mechanisms reduce the efficacy of traditional stop-update-test-redeploy-restart approaches to the integration of new or modified services. Self-adaptive applications have been recognized as viable solutions for dealing with systems where size and complexity increase beyond the ability of humans to respond manually, coherently and timely to environmental and system changes [5]. Massive reuse of services and frequent updates of implementations corresponding to compatible interfaces are typical of service-oriented applications, and allow to define efficient domain specific self-adaptive solutions for service-oriented applications [1, 3].

In this paper, we present a self-adaptive approach for service-oriented applications that combines novel techniques into a traditional sense-plan-act control loop, where the subject system is connected to a controller that in turn feeds commands back into the subject system.Our control loop works as follows: The invocation of a service triggers *monitoring* mechanisms. Such mechanisms identify changes in the invoked services that may depend on server-side implementation updates or dynamically discovered services. The detection of a new service implementation triggers *diagnosis* mechanisms that run test cases on the target service to reveal possible mismatches. If any mismatch is revealed, *adaptation* strategies update structure and behavior of the client to solve the identified problems, and optimize the interaction with the service.

The sense-plan-act control loop is instantiated for each set of services that comply with a specific contract. The customization consists of defining a set of test cases to reveal mismatches between different implementations of the same contract, and a set of adaptation strategies for the possible mismatches. For example, a self-adaptive mechanism for the treatments of persistency in different shopping carts can include the set of test cases, and adaptation strategies outlined in Figure 1. The control loop instantiated with test cases and adaptation strategies enhances client invocations of the cart service. The enhanced clients will intercept all calls to the cart service, dynamically check the persistence of the cart, and adapt the local behavior accordingly.

3 Experience Report

In this section, we outline a prototype implementation of the self-adaptive approach used in our experiments. The approach consists of a pre-processing and a generation step.

Fourth International Conference on Autonomic Computing (ICAC'07)
0-7695-2779-5/07 $20.00 © 2007 IEEE

Mismatch	Test cases that...	Adapters that...
Value interpretation	return distinguishable results	convert values
Violation of value domains	use representative domain values	filter calls and handle violation
Unexpected side-effects	return perceivable side effects	mask or fix the side-effect
Missing or misunderstood function	depend on the considered function	fix the function
Violation of non-func. req.	reveal the non-functional profile	meet the non-functional profile
Interaction protocol	allow identifying protocol models	map expected and actual protocols
Input cardinality	identify alternatives input conf.	map between different input conf.

Table 1. Excerpt of the mismatch checklist

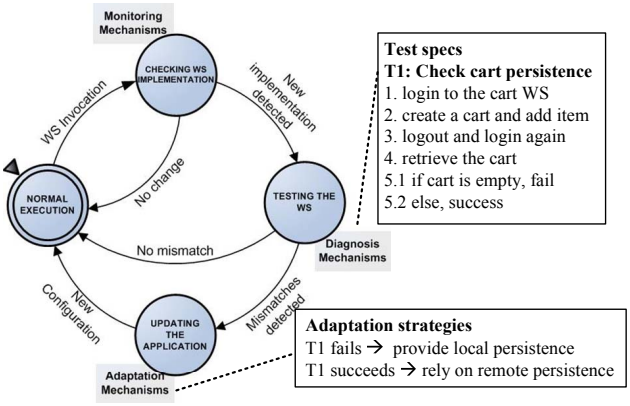

Figure 1. Control loop for cart persistency

During the pre-processing step, service integrators use an integration mismatch checklist to identify possible integration mismatches, generate test cases for revealing integration problems, and design suitable recovery actions. An excerpt of the checklist used in the experiments is shown in Table 2. During the generation step, a prototype tool deploys test cases and adaptation strategies in separate modules, and weaves these modules into the client applications. Further details about the prototype are described in [4].

Here we report the results of using the prototype for designing three sample self-adaptive applications that use third-party web services: *World-Weather*, a SOA-based application that provides statistical data about weather conditions, using web services to collect data in selected locations; *Virtual-Store*, a SOA-based web portal that facilitates comparisons and purchases across a set of e-commerce applications; the *Personal-Mobility-Manager (PMM)*, a SOA-based advanced navigation system ([6]). Table 2 shows the statistics of our approach on these baseline applications.

Application	services	mismatches	test cases	adapters
World-Weather	1	1	1	1
Virtual-store	3	11	24	11
PMM	7	24	66	24

Table 2. Statistics of the experiences

These initial experiences highlight several important aspects of our approach. First, from the design viewpoint, we found extremely beneficial focusing on functional and adaptation concerns at different times. Second, much of the infrastructure for controlling the dynamics of sense-plan-act loops can be automatically generated by our prototype, thus eliminating many technical difficulties. Next, the fault taxonomy appears as the natural joint point for integrating domain knowledge in the approach. Finally, the loose dependency between adaptation modules and client applications suggests the possibility of designing libraries of standard adaptation modules.

The preliminary experimental results indicate the efficiency of the self-adaptive approach to overcome non-trivial mismatches. The degree of self-adaptability of an application depends on the quality of the fault-taxonomy that guide the mismatch identification process. We are currently conducting extensive experiments on industrial benchmarks to further investigate the benefits of the approach, and improve our prototype.

References

[1] L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. In *Proc. of the Int. Conf. on Service Oriented Computing (ICSOC)*, 2005.

[2] D. Beyer, A. Chakrabarti, and T. Henzinger. Web service interfaces. In *Proc. of the Int. World Wide Web Conf. (WWW)*. ACM Press, May 2005.

[3] C. Dabrowski and K. Mills. Understanding self-healing in service-discovery systems. In *Proc. of the Workshop on Self-healing systems*. ACM Press, 2002.

[4] G. Denaro, M. Pezzè, and D. Tosi. SHIWS: a self-healing integrator for web services. In *Proc. of ICSE - Research Demos*. ACM Press, May 2007.

[5] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, Jan 2003.

[6] D. Lorenzoli, S. Mussino, M. Pezzè, D. Schilling, A. Sichel, and D. Tosi. A SOA-based self-adaptive personal mobility manager. In *Proc. of the IEEE Conf. on Service Computing (SCC Contest)*, Sept 2006.