Autonomic Business-Driven Decision Making for Adaptation of Web Service Compositions

Qinghua Lu1,2

Advisors: Dr. Vladimir Tomic1, Prof. Ross Jeffery1,2

1School of Computer Science & Engineering, University of New South Wales, Sydney, Australia
2Business Adaptation and Interoperation Project, NICTA, Sydney, Australia

Qinghua.Lu@nicta.com.au

Abstract—Runtime adaptation of Web service compositions can usually be done in several ways, so it is necessary to decide which adaptation approach to take. The PhD research presented in this paper provides a novel decision making approach, new management algorithms, and a middleware architecture for runtime adaptation of Web service compositions in ways that maximize business value, while satisfying all given constraints. All necessary information about possible adaptations and their business metrics are specified as policies in the WS-Policy4MASC language and the optimization problem is modeled in the powerful constraint programming language MiniZinc. The decision making algorithms integrated into the MiniZnMASC middleware allows it to determine how to adapt each Web service composition instance so the overall business value is maximized, while satisfying all given constraints (e.g., about resource limitations). Experiments with the MiniZnMASC prototype showed that the new solutions are feasible, functionally correct, business beneficial, with low performance overhead, and with linear scalability.

Keywords—Business-driven IT management; distributed system management; runtime adaptation; autonomic computing; service-oriented computing; constraint programming

I. INTRODUCTION

A. Problem Description

Service-oriented architecture (SOA) implemented with Web service technologies [1] is currently the most popular approach to build information technology (IT) systems for business processes. When business and technical changes occur in running Web service compositions implementing complex, long-running business processes, adaptation can usually be done in several ways and decision making is needed to determine how to proceed with the adaptation. In the business-driven IT management (BDIM) [2] research area, decisions are made based on mappings between business and technical metrics and are performed to maximize business value for the enterprise. Autonomic computing [3] is an approach where IT systems self-manage themselves using configurable policies. A policy specifies a collection of high-level, implementation-independent, operation and management goals and/or rules. Autonomic BDIM is the intersection area of autonomic computing and BDIM, which adds processing of business metrics to decision making components of autonomic systems.

While there has been a lot of research in the area of adaptation of Web service compositions and business processes, there has been relatively little research on advanced support for business-driven decision making for such adaptation, particularly when several Web service composition instances should be adapted at the same time, with minimal human intervention (i.e., autonomically). The main question my PhD research addresses is how to make decisions for runtime adaptation of Web service compositions in ways that maximize business value, while satisfying various constraints. This main research question includes four sub-questions:

- What are appropriate optimization algorithms for instance-specific adaptation decision-making?
- How to model the instance-specific adaptation decision-making problems?
- What policy language constructs are useful for operation of the newly developed algorithms?
- How to design and prototype software that uses the above principles?

B. The Proposed Approach and the Expected Contributions

My PhD research proposes new problem models, algorithms, architectures, and specification constructs that are useful for building middleware that makes adaptation decisions that improve overall business value. The following are the expected contributions to knowledge:

- Classifications of problems in decision making for business process adaptation (published in [4]).
- Problem models: My PhD research discovers how to represent the adaptation decision-making problems in constraint programming (published in [5]).
- Decision making algorithms: I develop different autonomic business-driven decision making algorithms for business process adaptation, including a basic BDIM algorithm (published in [4]), an extended algorithm using the Business Motivation Model (BMM) [6] information, an algorithm for concurrent adaptation of multiple instances (published in [5]), and an algorithm for adaptation of RESTful business processes.
- Policy language constructs: My PhD research examines which policy language constructs (e.g., expressing adaptation alternatives, business values, business strategies) are necessary (or useful) for operation of the newly developed algorithms. When a useful concept is

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.
not present in the policy language WS-Policy4MASC [7]. I add new constructs to this language.

- Middleware architecture: I design, prototype and evaluate middleware MiniZnMASC [5] that uses the above principles. The most important part of the middleware is the Policy Conflict Resolution module, which implements business-driven adaptation decision making algorithms that decide which of the adaptation alternatives should be executed. Its inputs are policies and runtime data, while its outputs are adaptation decisions.

C. Significance of the Research

Business metrics (such as profit and customer satisfaction) are more important to business users than technical metrics, such as availability and response time. The majority of past works focus on maximization of technical metrics, but this might not lead to maximization of business metrics. The solutions developed in my PhD research focus explicitly on maximization of various business metrics (both financial and non-financial ones). Further, in different business circumstances and strategies different business aspects are important (e.g., whether to maximize shorter-term or longer-term business metrics). The developed algorithms can make adaptation decisions considering various aspects. The algorithms can also concurrently make different adaptation decisions for different classes of Web service composition instances in a way that achieves globally optimal total business value while satisfying all given constraints. RESTful implementation of business processes [8] gains popularity, but their runtime operation can require complex decisions. The new algorithm for adaptation of RESTful business processes can determine which process fragment to execute for particular user in RESTful business process systems when multiple process fragments are available in a decision making point. Since the evaluation of the proposed solutions shows business benefits [5], the application in practice could save companies significant amount of money and increase customer satisfaction.

D. The Research Methodology

My PhD research started with a classification of problems in decision making for adaptation of Web service compositions, based on work in related literature and realistic motivating scenarios. This classification was published in [4]. It concluded that various aspects related to autonomic BDIM decision making are unexplored, yet very important from the business viewpoint. I also selected several specific sub-problems (e.g., concurrent decision making for multiple Web service composition instances, application to RESTful business processes) as the main focus of my research.

After selecting the research focus, I explored how to build better models (e.g., using constraint programming) for the selected problems. I have been developing different autonomic business-driven decision making algorithms for adaptation of Web service compositions and used the new problem models in these algorithms. Some of these algorithms were published in [5] and [4], but I still work on additional ones. I also examined which policy language constructs are necessary for operation of the newly developed algorithms and designed a new middleware that demonstrates practical operation of these algorithms.

The proposed solutions have been evaluated in five aspects: feasibility, functional correctness, business benefits, performance overhead, and scalability. Feasibility is evaluated through implementation of proof-of-concept prototypes. Functional correctness is evaluated by comparing results calculated by the developed algorithms and by hand, on several examples. To verify that the solutions found by my PhD research are always better or equal from the business viewpoint compared to the other possible solutions, several examples were developed to calculate the total business value and cost of solutions generated by different algorithms for adaptation selection. Performance and scalability is evaluated by measuring the performance of the middleware with increasing number of adaptation options and business value metrics. Apart from [5] and [4], there will be additional publication of evaluation conclusions.

II. RELATED WORK

The management of business processes and their service-oriented implementations brings a number of challenges because of multiple domains of ownership. Many research projects are working on this area. Dynamo [9] is a self-healing framework for BPEL processes, which is based on two languages: WS-CoL and WS-ReL. WS-CoL defines runtime constraints for BPEL processes while WS-ReL specifies recovery strategies. Dynamo is focused on monitoring Web service compositions while my PhD research is more focused on adaptation, taking into account the business metrics. AO4BPEL [10] extends BPEL with aspect-oriented concepts, which weaves aspects into Web service compositions to manage technical QoS metrics. In addition to QoS, My PhD research takes business value into account. Also, policies provide better abstraction for specifying management information than aspect advices.

The past business process adaptation decision making algorithms (e.g., [11]) predominantly maximized technical metrics, while maximization of business metrics is still an open research area [7]. On the other hand, [12] is not directly on adaptation of business processes, but some of its solutions could be reused in my PhD research. It presented a system for maximization of business metrics that schedules triggered management policies by minimizing the penalty specified in service level agreements, but it did not examine resolution for conflicting policies, which is important in policy-based management and one of the main topics of my PhD research. My PhD research is different from traditional optimization problem in business processes. My approach focuses on the decision making support for concurrent runtime adaptation of multiple Web service composition instances to maximize business value.
III. RESEARCH PROGRESS

A. Modeling Adaptation Decision Making in Constraint Programming

The initial efforts concentrated on representing the decision making model for concurrent adaption of multiple Web service composition instances in constraint programming. The notations are as follows. N specifies the number of classes of instance, across all supported business process types. \( X_n \) represents the current number of instances in class \( n \). \( M_n \) represents the number of adaptation options for an instance in class \( n \). \( A_{n,i} \) represents the number of business value types that can be reasoned about. \( W_k \) represents the weight of business value type \( k \). Weights are usually in the interval \([-1,1]\) (negative values are for costs), but can have other values. \( B_{n,i} = \sum_{k \in \text{UsedBVTs}} (W_{V_{k,n,i}}) \), represents the summary business value of all used business value metric types for adaptation option \( A_{n,i} \). The set \( \text{UsedBVTs} \) contains all business value metric types that are deemed relevant for comparing business worth of adaptation options. The problem of finding the globally optimal set of adaptations can be represented in constraint programming as the task to find the set of \( N \) adaptations \( j_n \) (class of instance), to maximize and satisfy additional constraints. These additional constraints can be of different types, but the research emphasizes cost limits, resource limits, and risk tolerance, as the most important in practice. Further details can be found in [5].

![Figure 1. Architecture of the management system for business processes and their service-oriented implementations](image)

B. Architecture of MiniZnMASC

Fig. 1 shows the architecture of the designed autonomic management system for business processes and their service-oriented implementations [5]. There are 3 groups of components and artifacts: the extended systems (incl. the managed system), MiniZnMASC, and file creation components. On the left-hand side of Fig. 1 are optional components and artifacts in our system. Their main purpose is to help users in creation of WS-Policy4MASC files, WS-PolicyAttachment files, and MiniZinc model files. On the right-hand side are components that MiniZnMASC system extends and integrates with, without re-implementing them: the executing managed system, third-party modules for monitoring the managed system, and third-party modules for executing chosen adaptation actions.

MiniZnMASC decides adaptation actions from the business viewpoint, based on monitoring data about execution of the managed business processes and according to policies in WS-Policy4MASC [7]. The information guiding adaptation decisions is specified in WS-Policy4MASC policies, consisting of policy assertions. WS-Policy4MASC policies are parsed by the Policy Parsing module and then stored in the Policy Repository module. Database of Monitored Data stores runtime data about monitored technical metrics, business metrics, and events that come from the external monitoring modules, after this data is handled by the Processing of Monitored Data module. Based on the recent monitored data and historical information stored in the Database of Monitored Data, the Determining Triggered Policies module determines which WS-Policy4MASC policy assertions should be executed.

The most important part of MiniZnMASC is the Policy Conflict Resolution module, which implements the adaptation decision algorithms that decide which triggered policy assertions (each representing an adaptation option) should be executed. The business-driven decision making model in constraint programming is implemented in the MiniZinc constraint programming language [13] and processed by the MiniZinc solver located in the Selection among Alternative sub-module. To select the globally optimal array of adaptations, the MiniZinc solver populates the given MiniZinc model with 2 types of runtime data: 1) total business values
(incl. non-financial values modeling resource constraints) and total costs for particular adaptation options, 2) the current number of running instances of each class of instance. The first type of runtime data is sent by the Calculation of Business Metrics sub-module, which uses the algorithm published in [4] and WS-Policy4MASC policy assertions stored in the Policy Repository. The second type of runtime data is stored in the Database of Monitored Data, based on notifications from external monitoring modules about starting/stopping of instances.

Since MiniZinc is much slower than Java in calculations of business value metrics, I have adopted the architecture in which calculation of business value metrics is done by fast Java code, while MiniZinc models deal with complex decisions for which Java (or other programming language) code would be much more complicated and harder to maintain. Another important benefit of such integration is that it significantly simplifies MiniZinc models, which now depend on the content of WS-Policy4MASC files only to a limited extent because policy conflict situations have common characteristics captured in a generic model I developed. These simpler MiniZinc models require less MiniZinc expertise from software engineers who develop them.

C. Application of MiniZnMASC to RESTful Business Processes

In RESTful business process [8], decision making points are positions in the business process where a set of process fragments are provided from which the next step is chosen. I developed new algorithms determining which process fragment to execute for particular users in RESTful business process systems when multiple process fragments are available in a decision making point. These selections depend on business metrics and business strategies, plus operational conditions. This application introduced several kinds of monitoring resources into RESTfulBP to provide the monitoring functionality. The monitored runtime data is delivered to MiniZnMASC in an XML document to be used for the adaptation decision making. The adaptation action is invoked by sending a HTTP request to the RESTfulBP engine, which performs the adaptation execution.

D. Evaluation and Benefits

After implementing several proof-of-concept prototypes of the proposed solutions (e.g., MiniZnMASC middleware), no problems with feasibility of the proposed solutions were found. I also implemented several realistic examples and evaluated the functional correctness by comparing the results calculated by MiniZnMASC and by hand. These results show that MiniZnMASC was built correctly. To verify that the solutions found by MiniZnMASC are always better or equal compared to the other possible solutions, I extended the tests of functional correctness to calculate the total business value and cost of solutions generated by different algorithms for adaptation selection. The results show that the decision making algorithms in MiniZnMASC have business benefits over other decision making algorithms (e.g., those maximizing technical metrics). For the performance and scalability tests, I measured the performance of MiniZnMASC with increasing number of adaptation options, used business metrics, and classes of Web service composition instances. The overall execution time of decision making in MiniZnMASC increases almost linearly with the problem complexity, showing good scalability, and in realistic uses never exceeds acceptable limits. Further details are discussed in [4-5].

IV. REMAINING OBJECTIVES

BMM [6] is an industrial standard for specification of high-level business motivation and intent as input into design, development and execution of IT systems. The next step of my PhD research is to add BMM support, to MiniZnMASC so that MiniZnMASC can make adaptation decisions taking long-term business considerations into account.

Another future work is to design the Policy Learning module of MiniZnMASC, in which policies specified in WS-Policy4MASC can be modified during runtime based on the monitored data using machine learning technologies.

REFERENCES