Finding cost-efficient adapters

Christian Gierds*

Humboldt-Universität zu Berlin, Institut für Informatik, 10099 Berlin, Germany
gierds@informatik.hu-berlin.de

Abstract. When adapting services in a SOA environment, not only the validity of the adapter may be of importance, but also non-functional properties like the costs of the adapter. We introduce an approach for finding cost-efficient adapters based on the operating guideline, which characterizes all valid adapters for the given services.

1 Introduction

In the context of *Service-Oriented Architectures* (SOA) [1] composition of actually incompatible services, which have a well-defined interface in order to offer a special functionality, is highly demanded. A service of one organization may not have been designed to work together with a service of a different organization. But before changing one or even both of these services, an appropriate adapter may help to overcome the incompatibility. A (behavioral) adapter then acts between the two different services and controls their communication in such a way, that a certain set of functional properties like deadlock freedom or weak-termination is satisfied.

Especially in corporate environments costs like time, memory or money are relevant factors for components. So if a company decides to use an adapter, it may want that the overall runtime of the adapter stays below a certain limit in order to guarantee some real-time constraints or the costs for using third parties should be minimized. Besides this demand, the adapter still needs to be valid – the original goal to resolve incompatibilities must be maintained.

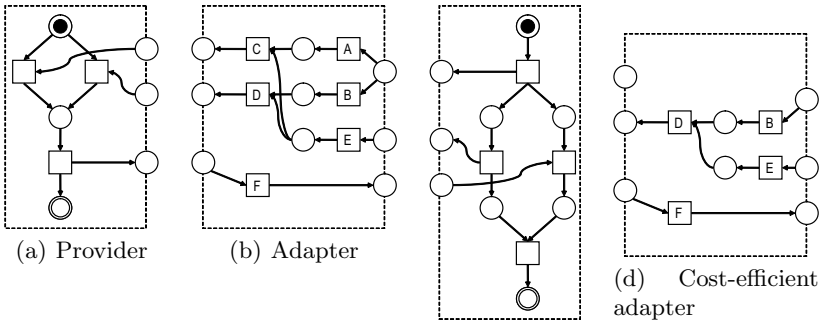
Our approach focuses on the minimization of the most expensive run of an adapter, meaning that for every other valid adapter the most expensive run is at least as expensive as for the calculated adapter.

The paper is structured as follows: In Sect. 3 we will describe the approach which covers both the validity and the cost optimization of adapters. Before it, we will introduce the basic formalisms in Sect. 2, namely open nets and operating guidelines. In the last section, we will summarize the obtained results and give an outlook on extensions of this approach.

2 Adapting services

An adapter is an artifact acting as mediator between services. This is necessary if the adapted services are incompatible regarding their interface or their behavior.

* Supported by German Research Foundation (DFG) under grant RE 834/18-1



(a) Provider (b) Adapter (c) Requester (d) Cost-efficient adapter

Fig. 1. Example

The adapter then should overcome these incompatibilities and guarantee a well behaved interaction of the services.

There are different approaches for adapters like [2–6] and recently also [7], to which we will refer. They mainly differ in the way, how elementary actions of an adapter are described and derived, and how the actual adapter is calculated.

In [7] a two-step approach is presented. First, for two given services P and R a *rewriter part* E (part of the final adapter) connects to the interfaces of P and R, provides transitions to manipulate messages based on a set of simple rules, and creates an interface for triggering these transitions. In the second step, a *controller part* C for the composition of $P \oplus E \oplus R$ is calculated such that certain properties like deadlock freedom etc. are ensured (see [8]). The suggested cost optimization in this paper is executed on C.

Open nets The used adapter approach is based on *open nets*, an extension of Petri nets, where distinguished places act as interface.

Definition 1. *The tuple* $(P, I, O, T, F, m_0, \Omega)$ *is called an* open net *iff*

- (P, T, F, m_0) *is a Petri net with a set of places* P*, a set of transitions* T*, a flow relation* $F \subseteq P \times T \cup T \times P$ *and an initial marking* $m_0 : P \to \mathbb{N}$*,*
- I *and* O *are disjoint sets of input* $I \subseteq P$ *and output* $O \subseteq P$ *places,* $I \cap O = \emptyset$*, and*
- Ω *is a set of final markings.*

Two open nets can be composed ($\oplus$) by merging equally named input and output places. The firing rule is equivalent to the one of regular Petri nets. A marking is not called a deadlock if it is included in the set of final markings.

The nets depicted in Fig. 1 are open nets. The places on the dashed border form the interface, all places belonging to the initial marking contain a black

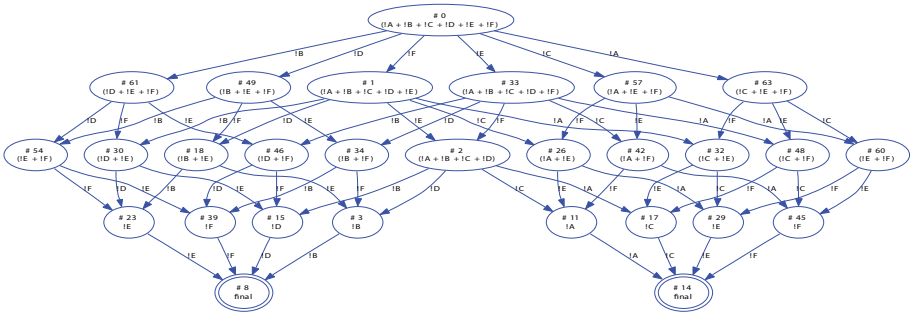


Fig. 2. Operating Guideline for Adapter in Fig. 1(b)

token and places belonging to a final marking are shown with a two line border. If the nets in Fig. 1 are composed as implied by the figures' alignment, the composition will be deadlock free.

Adapters The open nets P and R represented in Figs. 1(a) and 1(c) are not compatible, since the number of exchanged messages does not fit. Based on certain message transformation rules provided with the two services an adapter like Fig. 1(b) might be build based on the considerations in [7].

In our example let `A-F` be such rules that transform messages. Instead of executing these rules arbitrarily, we provide an interface such that these rules can be triggered by an controller. This controller will ensure certain properties like deadlock freedom, if wanted (see [11]). In the following, message exchange will be called an *event*.

Figure 2 shows such a controller. The graph, called *Operating Guideline*, is a labeled transition system, where each edge label represents an event, namely the sending or receiving of a message. Furthermore each node n has an annotation Φ over the labels of the edges leaving n. A Φ satisfying assignment β corresponds to a valid combination of edges, that have to be included in an controller, such that the controller is valid. In this paper we assume the operating guideline to be acyclic.

The controller part C can be transformed to an open net using the approach in [12], such that C and E can be composed to form a valid adapter $E \oplus C$.

3 Cost optimization

Looking at the possible adapters for P and R, besides some control structures we can mainly distinguish each single adapter by the transformation rules it can execute. It is legitimate to assume, that these transformation rules generate the costs in a corporate environment. In the simplest case, time is consumed to apply such a rule. In a more distributed scenario, such a transformation might

be done by an external service provider, which will result in a fee which must be paid.

Just eliminating expensive transitions in the adapter is not an solution for finding cost-efficient adapters. When changing the adapter, we have to ensure that the correctness criterion like deadlock free communication is maintained. Therefore approaches like [9, 10] are not usable in this scenario, since we do not want to just calculate the cost for a service, but based on cost estimation build a service, in this case an adapter.

The operating guideline calculated during the adapter synthesis contains all information about legal adapters. Therefore an cost optimization should be done on this structure. There are two points we have to take care of, namely *i)* the most expensive run of the adapter shall be minimized and *ii)* the resulting structure must still be a valid adapter (thus, $P \oplus A \oplus R$ still must be deadlock free).

If we look at any given adapter A, it is still possible that A has different execution traces, i.e. sequence of events, depending on internal decisions in the services P and R.

The costs for one run of the adapter, a trace of events, is simply the sum of each applied message transformation. Thereby the cost function is a mapping of every transformation rule $r \in R$ to a natural number: $c : R \mapsto \mathbb{N}$.

Definition 2. *The* costs of a trace t *is the sum of its contained events:* $c(t) = \sum_{i=1}^{k} c(r_i)$ *for* $t = (r_1, \ldots, r_k)$.

We will focus on the question, which is the worst case, meaning, which are the highest costs we have to anticipate regarding the possible traces of A.

Definition 3. *The* costs of an adapter A *is the maximum costs over all traces of* A*:* $c(A) = \max_{t \in traces(A)} \sum_{i=1}^{k} c(t)$.

Based on Def. 3 our optimization goal is to find an adapter, whose costs are at most as expensive as for any other valid adapter.

Definition 4. *An adapter* A *is* cost-efficient *if for any other adapter* A' *yields* $c(A) \leq c(A')$.

The controller introduced in the previous section contains all information necessary for finding cost-efficient adapters. The annotations provide details about which other controllers are valid, and since every application of an transformation rule is communicated, it also contains all execution traces as a branching structure.

In order to find an cost-efficient adapter we will annotated each edge with a set of traces as follows. Given these edge annotations, we will compute an assignment for each node's annotation.

The costs incurred by using an edge is the maximum (the worst case) costs of the traces that are possible via this edge.

Definition 5. *The* costs of an edge e *is the maximum cost of its assigned traces:* $c(e) = \max_{t \in traces(e)} c(t)$.

Given a satisfying assignment β for the annotation of a node n, β states which edges leaving n have to be included in a controller in order to be valid. The node's costs regarding β then is the maximum over the edges' costs (again, the worst case).

Definition 6. *The* costs for a node n and an assignment β_n *is the maximum cost of the edges* e *leaving* n*, which corresponding literal is set to true in* β_n*:*
$c_{\beta_n}(n) = \max_{\beta_n(e)=true} c(e)$.

Since every satisfying assignment yields in a valid adapter, we choose an assignment, which results in the minimal costs for a node.

Definition 7. *The* costs for a node n *is the minimum costs over all assignments* β_n *satisfying* n*'s annotation* Φ*:* $c(n) = \min_{\Phi(\beta_n)=true} c_{\beta_n}(n)$.
The assignment β_n*, which minimizes* $c(n)$ *is called the* minimal β_n.

Given the previous definitions the following algorithm will calculate a cost-efficient adapter.

Algorithm Let P and R be two open nets, E a partial adapter, and C the operating guideline for $P \oplus E \oplus R$. Then an *cost-efficient adapter* can be found as follows:

Initially all edges have no traces assigned, and all nodes have infinite costs, except for the leaf nodes (without successor), which have costs 0 (resulting from Def. 7). Then, as long as there are nodes with infinite costs, pick such a node n, so that each successor n' of n has finite costs $c(n') < \infty$. Assign for each edge $e = (n, n')$ the set of traces according to the minimal $\beta_{n'}$ of n': $traces(e) = \{label(e) + t' \mid t' \in traces(e'), \beta_{n'}(e') = true\}$ (meaning: each new trace starts with $label(e)$ followed by the events in the traces of e'). Afterwards the costs $c(n)$ can be calculated.
The costs for resulting adapter $A = E \oplus C$ are the costs of C's root node.

Since we assume the operating guideline to be finite and acyclic, it can be easily seen, that the suggested algorithm will terminate, and all nodes will have finite costs. Furthermore we gain a valid controller (implied by the found minimal assignments), which minimizes the costs.

Theorem 1. *The provided algorithm finishes with a controller* C *such that* $A = E \oplus C$ *is a cost-efficient adapter for* P *and* R.

Proof. (Sketch.) This result yields mainly due to Def. 7. The adapter A is valid, since for each node n of the controller, the minimal β_n satisfies n's annotation (see [8] for details). Assume A' is another valid adapter with less costs $c(A') < c(A)$. Since both A and A' are derived from C there exists a node n included in both adapters, but differing in the minimal β_n, meaning $c_{A'}(n) < c_A(n)$, which contradicts Def. 7. Therefore the found adapter is valid and cost-efficient.

4 Summary

We have seen an approach which calculates a cost-efficient adapter based on an annotated graph which acts as controller for the application of the message transformation rules. By finding optimal assignments to the nodes' annotations we obtain a smaller, but valid adapter, where expensive runs can be excluded.

The time for finding such an adapter is exponential in the number of applicable message transformation rules in the worst case, since for every node all satisfying assignments must be checked. Nevertheless in most cases the approach should yield the result in a reasonable time, since the annotations are normally short and therefore quickly to be checked. In order to show its feasibility, this algorithm shall be implemented and checked with real-world examples.

As extension of this approach probabilities for the occurrence of events shall be introduced, so that the *average* costs of an adapter can be calculated. This extension would also allow to lift the optimization to cyclic controllers. Although cyclic adapters are finite, the have however infinite traces and therefore infinite costs.

References

1. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson - Prentice Hall, Essex (July 2007)
2. Brogi, A., Canal, C., Pimentel, E.: On the semantics of software adaptation. Science of Computer Programming **61** (2006) 136–151
3. Benatallah, B., Casati, F., Grigori, D., Motahari Nezhad, H.R., Toumani, F.: Developing Adapters for Web Services Integration. In: Proc. CAiSE. Volume 3520 of LNCS. (2005) 415–429
4. Bracciali, A., Brogi, A., Canal, C.: A formal approach to component adaptation. Journal of Systems and Software **74**(1) (2005) 45–54
5. Brogi, A., Canal, C., Pimentel, E., Vallecillo, A.: Formalizing Web Service Choreographies. Electr. Notes Theor. Comput. Sci. **105** (2004) 73–94
6. Dumas, M., Spork, M., Wang, K.: Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation. In: Proc. BPM. Volume 4102 of LNCS., Springer (2006) 65–80
7. Gierds, C., Mooij, A.J., Wolf, K.: Specifying and generating behavioral service adapter based on transformation rules. Preprint CS-02-08, Universität Rostock, Rostock, Germany (August 2008)
8. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics **1**(3) (2005) 35–43
9. Hee, K.M.v., Sidorova, N., Stahl, C., Verbeek, H.M.W.: A price of service in a compositional SOA framework. Computer Science Report 07/16, Technische Universiteit Eindhoven, The Netherlands (jul 2007)
10. vom Brocke, J., Lindner, M.A.: Service portfolio measurement: a framework for evaluating the financial consequences of out-tasking decisions. In: ICSOC. (2004) 203–211
11. Schmidt, K.: Controllability of Open Workflow Nets. In: Enterprise Modelling and Information Systems Architectures. Volume P-75 of LNI. (2005) 236–249
12. Badouel, E., Darondeau, P.: Theory of Regions. In: Lectures on Petri Nets I: Basic Models. Volume 1491 of LNCS., Springer-Verlag (1996) 529–586