Creating Message Profiles of Open Nets

Jan Sürmeli and Daniela Weinberg

Humboldt–Universität zu Berlin, Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
{suermeli,weinberg}@informatik.hu-berlin.de

Abstract. In a network of services, external and internal decisions, asynchronous message exchange, and concurrency induce complex interaction protocols. In this paper we introduce the notion of a message profile of a service that is modeled as a special kind of Petri net.
The message profile is obtained solely from properties of the given net without requiring knowledge of interacting nets. It provides insight into the interactional behavior of the service. The information may then be used to enhance existing service analysis techniques as well as to verify the service model on a message basis.

1 Introduction

The central part of the evolving paradigm of Service-Oriented Computing (SOC) are services. A service represents a self-contained software unit that offers an encapsulated functionality over a well-defined interface. The promising goal of a SOC architecture is to ensure for each participating service to be loosely coupled with another service with little effort [1] and thus, creating a network of services, that is able to handle certain tasks. In contrast to other paradigms it is possible to create a heterogenous network that crosses organizational boundaries.

In general, a service is not designed to be used stand-alone. It is the stateful interaction of different services that adds significant value to SOC. Therefore, with respect to SOC, we are interested in whether every service instance will eventually terminate in a well-defined state with no useless (dead) activities being pending. This idea has already been formalized as usability in [2]. We use the term *controllability* instead of usability to avoid misunderstandings w.r.t. other well established meanings of "usability". We analyze whether two services S and S' can *interact properly* [3]. A service S' that properly interacts with service S is a *controller* of S. In our approach, we model a service as an open net [4, 5], which is a special class of Petri nets that extends classical Petri nets by an interface for communication with other open nets. We assume an asynchronous setting in which the order of sending messages does not necessarily correspond to the order of receiving those messages.

In a network of two or more services, external and internal decisions, asynchronous communication, and concurrency induce complex interaction protocols. In this paper, we introduce techniques that can be used to create a message profile of a given service S. This profile may then serve as a guide for a controller

C with respect to which messages may be sent to S as well as how often S accepts a particular message. The message profile is gained from analyzing the open net model of S. Thus, knowledge of C is not required. With the help of the behavioral properties stored in the message profile, we are able to characterize possible controllers. So, we can exclude certain controllers before-hand which enhances established methods such as partner synthesis [6] or the computation of the operating guidelines of S [5]. Furthermore, we enable the modeler of S to verify that the model mirrors its designated interactional behavior.

This paper is structured as follows. First, we briefly introduce open nets and controllability notions in Sect. 2. In Sect. 3, we present techniques to build up the message profile and show how it can be obtained by analysis of an open net. Finally, we conclude our results in Sect. 4.

2 Open Nets and Controllability

We model services with open nets [4], which enhance classical Petri nets. An open net is a tupel $N = (P, T, F, P_{in}, P_{out}, m_0, \Omega)$ with P being the set of places, T the set of transitions and F the flow relation. The set $P_{in} \subseteq P$ ($P_{out} \subseteq P$) represents the *input channels* (*output channels*) of the service. For the rest of this paper, we call P_{in} (P_{out}) *input* (*output*) *places* and $P_{in} \cup P_{out}$ the *interface* of N. For node $n \in P \cup T$ the set ${}^\bullet n = \{x \mid (x, y) \in F\}$ ($n^\bullet = \{y \mid (x, y) \in F\}$) is the *preset* (*postset*) of n. We demand that ${}^\bullet p = \emptyset$ ($p^\bullet = \emptyset$) for every $p \in P_{in}$ ($p \in P_{out}$) and $P_{in} \cap P_{out} = \emptyset$. Transition $t \in T$ with ${}^\bullet t \subseteq P_{in}$ / $t^\bullet \subseteq P_{out}$ / ${}^\bullet t \cup t^\bullet \not\subseteq P_{in} \cup P_{out}$ is called a *receiving* / *sending* / *internal* transition. m_0 is the initial marking and Ω is the set of final markings, which constitute the set of final states that the service should reach. The *inner* of a net N is obtained from N by removing the input/output places and adjacent edges from N. The set $LL = \{m_k, \ldots, m_n\}$ is a livelock iff all $m_i \in LL$ are mutually reachable and from no m_i an $m_j \notin LL$ is reachable.

Figure 1 depicts our example open net N1. The net has got five input places, namely a, b, c, d, e and one output place F. The initial marking of N1 is [p0] (depicted as a black token on place p0) and the set of final markings is $\{[p2], [p7]\}$. We can easily see that from [p0] N1 is not able to reach a different marking unless there is an additional token either on input place a, b or e. A token on an input/ output place represents a message. The open net N1 is able to send a message to the output place and to receive messages from its input places. Therefore, we are able to model the interaction between different open nets and thus have a formal notion for modeling the interaction of services.

The interaction of two different open nets N and C is expressed by their composition $N \oplus C$ which is obtained by merging every input place of one open net with the equally labeled output place of the other net (if that one is present).

Intuitively, controllability of an open net N means that N can *properly interact* with some other net. So, N is *controllable* if there exists an open net C, such that the composed open net $N \oplus C$ fulfills certain properties. Throughout

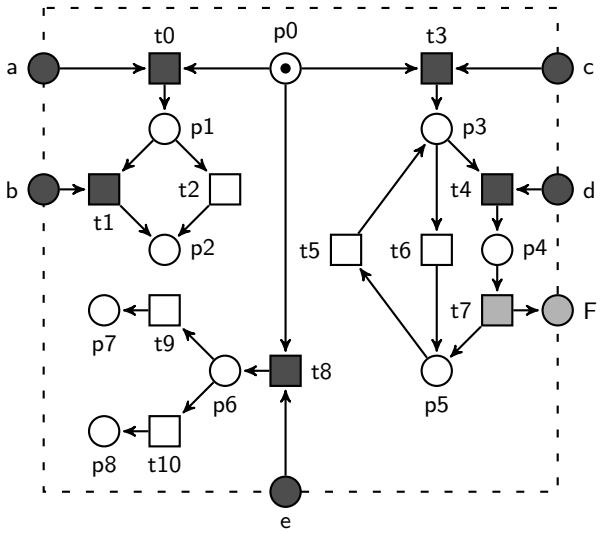

Fig. 1. Example open net N1.

this paper we will call C a *controller* of N. We distinguish between DF-, WT-, and RI-controllers based on the following notions [3].

Deadlock-Freedom (DF) states that all deadlocks of the composition $N \oplus C$ are final states of $N \oplus C$ and *Weak-Termination (WT)* (equal to *Livelock-Freedom*) specifies that from every marking of $N \oplus C$ a final marking of $N \oplus C$ is reachable. The property of open nets called *responsiveness* [3] can easily be mapped to the composition of two services – *Responsive Interaction (RI)*. A composition is responsive if either from every marking m of $N \oplus C$ a final marking of $N \oplus C$ is reachable, or from m a marking m' is reachable such that either N or C has sent or received a message. We further demand that each message sent will eventually be received.

Controllability is only decidable for those open nets whose reachability graph of the inner net is finite [7]. We also demand that the communication between two open nets is limited – there are no more than k messages ($k \in \mathbb{N}$) on any interface place at any reachable marking of the composition [5, 7].

3 Message Profile

In this section, we introduce techniques that analyze the net N in order to create a message profile of N. All methods avoid state space exploration and do not require knowledge of a controller C of N.

In the message profile we store different kinds of information – how often a specific message can be received, dependencies between messages, and which messages are not to be sent to N by a controller.

Intuitively, there are two requirements for an arbitrary service S to receive a message x: (1) S is in a state in which it accepts x, (2) a message x has been sent by controller C already. These requirements can easily be mapped to an open net N. Thus, (1) a receiving transition for x is enabled in the inner of N and (2) a token is available on the corresponding input place.

When analyzing a receiving transition t in the inner of N one less precondition for t to be enabled has to hold – the edge to its corresponding input place has been removed. Thus firing t does not depend on the number of tokens on that input place any more. So, we conclude that t can not fire more frequently in $N \oplus C$ than in the inner of N. Knowing how many times a receiving transition fires in the inner of N can thus lead to insights regarding the receiving behavior of N in $N \oplus C$. A *firing limit* for a transition t is a natural number n such that there exists no path in the reachability graph with t occurring more than n times.

There exist methods to compute the firing limit of a transition in a Petri net that avoid state space exploration. Currently, we are looking for an approach that fits best w.r.t. to the open net models of real processes. In this paper, usage of the term *firing limit* will refer to some firing limit determined by an arbitrary method. In the inner of our example open net N1 (Fig. 1) 1 is a firing limit for {t0,t1,t2,t3,t9,t10} and ∞ for transitions {t4,t5,t6,t7}.

We can use the firing limit for receiving transitions in the inner of the net as a basis and as an additional constraint for the creation of the message profile without requiring any knowledge about C.

3.1 Receiving Limit

The idea behind the *receiving limit* n of a message x is to determine how often x can be received by a service N at most, i.e. how often a receiving transition for x fires in $N \oplus C$. As we aim at being as general and controller-independent as possible, we make use of the firing limit for receiving transitions in the inner of N.

We calculate the receiving limit of message a of the example net N1 (Fig. 1). The only receiving transition for message a is t0. The firing limit for t0 is 1. Obviously, there is no guarantee that N1 will actually receive a, but it is safe to say that N1 will receive a only up to once. We set the *receiving limit* for a to 1.

Let us take a look at transitions t4 and t6 with ${}^\bullet\mathsf{t4} = {}^\bullet\mathsf{t6} = \{\mathsf{p3}\}$. Due to the cycle, the firing limits for both t4 and t6 are ∞. Thus, we cannot determine a finite number for the receiving limit of message d. So, we set it to ∞.

In N1 there exists exactly one receiving transition for each input place. Generally, for a message x, we set the receiving limit to the sum of the firing limits of *all* receiving transitions of x. If there exists no receiving transition for x, the receiving limit is 0 – N does not accept x in any marking. If we set the receiving limit for x to ∞, there are two possible reasons – (1) There exists no bound for N receiving x. Or, (2) we can not narrow down a finite number for such a bound. Either way, a receiving limit of ∞ does not allow further conclusions.

Finding a finite receiving limit n for a message x proves to be very useful. Although n might not be precise, as there is no guarantee that x will be received exactly n times, it is always safe to say that sending x more than n times leads to x being ignored. Therefore, we can conclude that every RI-controller respects the receiving limit of each message. Thus, we include it in the message profile. Summarizing, the receiving limits of the example net N1 are a:1,b:1,c:1,d:∞,e:1.

3.2 External one-time Decisions

So far, the message profile consists of a static number – the receiving limit for each input message, only. In an open net N, however, there might exist *dependencies* between messages that influence the actual acceptance of them. From the structure of N we are able to extract conflicting receiving transitions. The set of *external one-time decisions* contains all messages that these transitions receive, $\{x, y, z\}$. Basically, a controller C influences the further course of N by sending a message x, for instance. Because $\{x, y, z\}$ are in conflict, messages $\{y, z\}$ will now be ignored by N. We include such a set of messages into the message profile.

We will take a look at the open net N1 of Fig. 1 again. Examining the transitions t0, t3 and t8 in the inner of N1, we can easily see that there exists a conflict between them: $^\bullet$t0 = $^\bullet$t3 = $^\bullet$t8. The decision between t0, t3 and t8 is non-deterministic in the inner of N1. But, considering the interface, the choice is made by the controller – by sending one of the messages $\{\mathsf{a}, \mathsf{c}, \mathsf{e}\}$. Assume message a is received by N1. Then, no marking is reachable where one of $\{\mathsf{t0}, \mathsf{t3}, \mathsf{t8}\}$ is enabled again. Thus, sending message c or e would result in N1 ignoring that message.

In general, we construct such a set of messages M of an open net N as follows. We start by finding a set of receiving transitions that are in conflict with each other in the inner of N, set D. Then, we remove all transitions with a firing limit of 0 from D, set D'. From the receiving transitions in D' we extract the corresponding messages, set M. Set M does not necessarily reflect a global situation. Hence, we check whether the receiving limit of every message $m \in M$ is 1. This way we ensure that there exists no receiving transition $t \in T \setminus D$ for m. If that condition does not hold for a message, we remove it from M. We repeat that process until the condition holds for each message of M. The resulting set is now globally valid. So, sending more than one message from M always leads to N ignoring at least one message. Therefore, no RI-Controller sends more than one message from M which we include in the message profile. The set of conflicting messages $\{\mathsf{a}, \mathsf{c}, \mathsf{e}\}$ forms an external one-time decision of N1.

3.3 Internal Decisions

Internal decisions describe dependencies between receiving transitions and internal or sending transitions. They potentially induce that messages cannot be received. Based on the receiving limit we can decide if such a conflict leads to ignored messages or not.

We look at t1 and the internal transition t2 in N1 (Fig. 1). In the inner of N1, we find $^\bullet$t1 = $^\bullet$t2 = {p1}. Suppose marking m with m(p1) = m(b) = 1. The decision between firing t1 or t2 is non-deterministic. Thus, we call t1 *blocked* by an internal transition. The firing limits are t1:1,t2:1. Once $t \in \{\mathsf{t1}, \mathsf{t2}\}$ fires, no marking m' is reachable such that any $t' \in \{\mathsf{t1}, \mathsf{t2}\}$ is enabled. Hence, N1 decides non-deterministically between ignoring and receiving b .

For transitions t4 and t6 a similar pattern holds in the inner of N1, $^\bullet t4 = {}^\bullet t6$. The difference is, that whenever one of $\{\mathsf{t4}, \mathsf{t6}\}$ fires, a marking will be reached in which both transitions are enabled again. Assume now a message d is sent to N1 – even if t6 is fired consecutively, d can still be received in any reachable marking.

More generally, a receiving transition t is *blocked* by an internal or sending transition t' if $^\bullet t' \subseteq {}^\bullet t$ holds in the inner of N. We call a a *message x blocked* by an internal decision if (1) we find a finite receiving limit for x and (2) each receiving transition for x is either blocked or has a firing limit of 0. No RI-controller sends x. DF-controllers might send x, but then the composition will always contain a livelock. We include an according set of messages in the message profile. For N1 only message b is blocked by an internal decision, since condition (1) does not hold for message d.

3.4 Trap Messages

Deficient modeling, modification of an existing service or deliberate design of error conditions can lead to a structure, where the postset of an internal place (neither an input nor an output place) of the underlying open net N is either empty or consists of transitions with a firing limit of 0 in the inner of N. If such a place is not marked in any final marking, firing a transition in its preset *traps* N in a state from which no final state is reachable. We show under which circumstances messages can be tagged as *trap messages* in the message profile.

Examining place p8 in the example open net N1 (Fig. 1), we notice that from any marking m with m(p8) > 0 there will be no marking m' reachable with m'(p8) = 0. Additionally, m_f(p8) = 0 for each final marking m_f. Because of $^\bullet$p8 = {t10}, transition t10 should never fire. We call m a *trap marking*, p8 a *trap place* and t10 a *trap transition*. We propagate this property. Because $^\bullet$t10 = {p6}, we conclude that a token on p6 induces firing of t10 and thus to trapping the net. Thus, p6 is a trap place and all transitions of $^\bullet$p6 (= {t8}) are trap transitions. So, transition t8 is a trap transition. t8 is also a receiving transition for message e. Since there exists no other receiving transition for message e, it is obvious that sending message e will either lead N1 into a trap state (if t8 fires) or it leads N1 to ignore message e (if t8 does not fire). Therefore message e is not sent by any WT-controller.

To make sure that a specific message m is to blame for leading to a trap marking, we analyze its receiving transitions and check if each of them is either a trap transition or has a firing limit of 0. In that case, we tag m as a trap message in the message profile. In N1, only message e is a trap message.

4 Conclusion and Future Work

With the help of the example open net N1 (Fig. 1) we have shown that we are able to gain knowledge about the interactional behavior directly from the inner of N1 without knowing any controller C or building up the reachability graph of $\mathsf{N1} \oplus C$. We now know that any RI-Controller C sends messages a and c not more than once and completely avoids to send messages b and e. After having sent a message $x \in \{\mathsf{a}, \mathsf{c}, \mathsf{e}\}$, C sends no more messages from $\{\mathsf{a}, \mathsf{c}, \mathsf{e}\}$.

Currently, we work on a prototypical implementation of our results. We explore solutions for finding firing limits for transitions. Further, we improve our analysis methods and work on combining the methods to accomplish synergy effects. So far, we only focus on receiving messages. It is also possible to extend the message profile not only by further dependencies between receiving messages, but to include information about sending messages as well. Regarding that, we aim at developing a concept of compatibility of message profiles of two services in order to improve matching of two services.

References

1. Papazoglou, M.: Web Services: Principles and Technology. Pearson - Prentice Hall, Essex (2007)
2. Martens, A.: Analyzing Web Service Based Business Processes. In: FASE 2005. Volume 3442 of LNCS., Springer-Verlag (2005) 19–33
3. Wolf, K.: Does my service have partners? Transactions on Petri Nets and Other Models of Concurrency (2008) (Accepted for publication in November 2008).
4. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. AMCT **1**(3) (2005) 35–43
5. Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services. In: ICATPN 2007. Volume 4546 of LNCS., Springer-Verlag (2007) 321–341
6. Weinberg, D.: Efficient controllability analysis of open nets. In: WS-FM 2008. LNCS, Springer-Verlag (2008) accepted.
7. Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Inf. Process. Lett. (2008) accepted.