Wendy: a tool to synthesize partners for services

Niels Lohmann[1,2] and Daniela Weinberg[1]

[1] Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
[2] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
wendy@service-technology.org

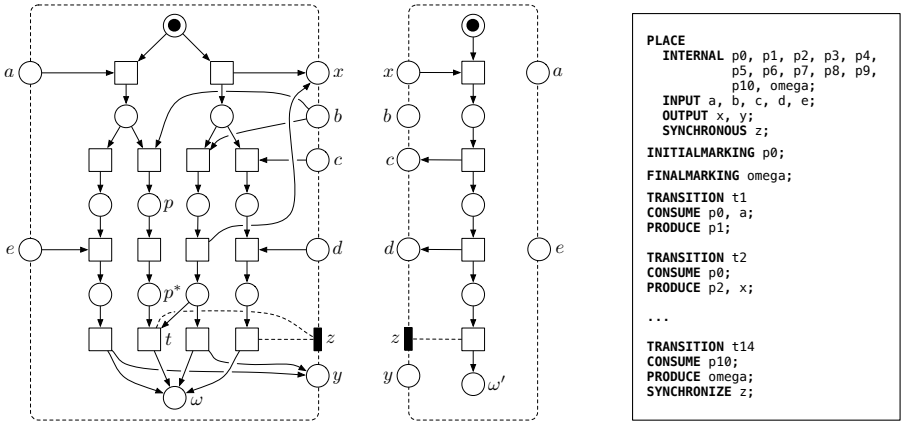
Abstract. Service-oriented computing proposes *services* as building blocks that can be composed to complex systems. To reason about the correctness of a service, its communication protocol needs to be analyzed. A fundamental correctness criterion for a service is the existence of a *partner service*, formalized in the notion of *controllability*.
In this paper, we introduce *Wendy*, a Petri net-based tool to synthesize partner services. These partners are valuable artifacts to support the design, validation, verification, and adaptation of services. Furthermore, Wendy can calculate an *operating guideline*, a characterization of the set of all partners of a service. Operating guidelines can be used in many application scenarios from service brokerage to test case generation. Case studies show that Wendy efficiently performs on industrial service models.

1 Objectives

The emerging field of service-oriented computing (SOC) proposes to build complex systems by composing services. A service encapsulates a certain functionality and offers it through a well-defined interface. As services are not executed in isolation, their communication protocol has to be considered when reasoning about the correctness of a service. To this end, *controllability* [1] has been introduced as a fundamental correctness criterion for services. A service is controllable if there exists a partner service such that their composition is compatible, for instance free of deadlocks. Controllability not only proves correctness of a service, but a partner service also provides insight into the communication behavior of the modeled service. Furthermore, it is a useful artifact to validate [2] or document the service and is the basis of adapter synthesis [3].

A controllable service usually has more than one partner service. An *operating guideline* finitely characterizes the (possibly infinite) set of all partner services of a service [4]. Operating guidelines are useful in a variety of applications including test case generation [5], service correction [6], instance migration [7], or service substitution [8]. They further allow to efficiently realize a service-oriented architecture in which the service provider publishes his operating guideline at a service broker. A service requester then only needs to check whether his service is one of the partner services that is characterized by the operating guideline [9].

In this paper, we introduce Wendy, a tool to synthesize partner services and to calculate the operating guideline of a service. Wendy provides the basis




```
PLACE
  INTERNAL p0, p1, p2, p3, p4,
           p5, p6, p7, p8, p9,
           p10, omega;
  INPUT a, b, c, d, e;
  OUTPUT x, y;
  SYNCHRONOUS z;

INITIALMARKING p0;

FINALMARKING omega;

TRANSITION t1
CONSUME p0, a;
PRODUCE p1;

TRANSITION t2
CONSUME p0;
PRODUCE p2, x;

...

TRANSITION t14
CONSUME p10;
PRODUCE omega;
SYNCHRONIZE z;
```

(a) open net N (b) partner N' (c) open net file format for N

Fig. 1: Open net N (a), a partner open net N' of N (b).

of a vast variety of applications which are essential in the paradigm of SOC. Case studies show that Wendy can cope with industrial and academic service models. We continue with sketching the functionality and the used formalism. The architecture of Wendy and the components it is built of are described in Sect. 3. Section 4 shows how Wendy can be used in different use cases, provides experimental results, and gives information about how to obtain and set up Wendy, before Sect. 5 concludes the paper.

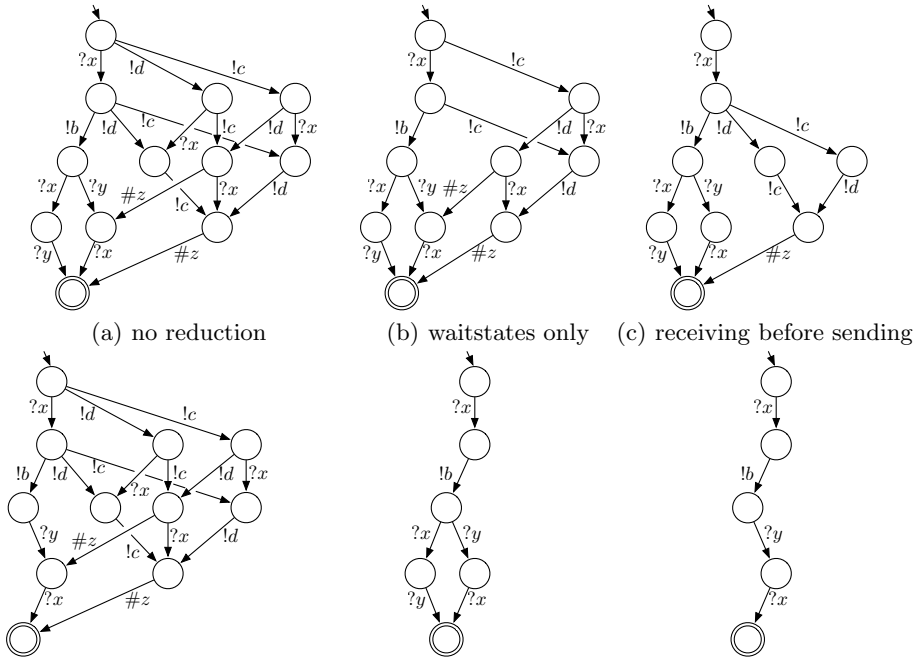
2 Functionality

2.1 Background

The theory [4,1,10] implemented by Wendy focuses on the behavior (both control flow and communication protocol) of a service. We model a service as an *open net* [11], a special type of Petri net with an interface. Open nets can be automatically derived from industrial service description languages such as WS-BPEL [12].

Figure 1(a) and 1(b) depict two open nets N and N'. The interfaces (modeling the message channels of the service) are depicted on the dashed frames. We distinguish asynchronous input and output message channels (modeled as places) and synchronous message channels (depicted as black rectangles). To distinguish desired final markings from unwanted deadlocks, an open net has a distinguished final marking ($[\omega]$ for N and $[\omega']$ for N'). We require the interface places to be empty in the initial and final marking.

The open nets N and N' can be composed by merging the input places of N with the output places of N' (and vice versa), and by fusing each pair of transitions of N and N' that are connected to the same synchronous channel. Initial and final markings are added element-wise. The composition $N \oplus N'$ is compatible, because the only reachable deadlock $[\omega, \omega']$ is a final marking. If N



(a) no reduction (b) waitstates only (c) receiving before sending

(d) sequentialize receiving events (e) succeeding sending event (f) quit as soon as possible

Fig. 2: Different types of partners of N by applying a certain reduction rule.

and N' are compatible, we call N as well as N' controllable, and refer to N as a partner service of N', and vice versa.

2.2 Partner synthesis

Wendy analyzes controllability of an open net and synthesizes a partner as a witness if the net is controllable. This partner is an automaton model that can be transformed into an open net using known tools such as Petrify [13]. In the automaton representation, asynchronous send actions, asynchronous receive actions, and synchronous actions are preceded by "!", "?", and "#", respectively.

The open net N is controllable, and Fig. 2(a) shows the partner synthesized by Wendy. By design, this partner is *most-permissive* in the sense that it simulates any other partner including N'. This partner reveals that no compatible partner of N will ever send an a-message. The transition connected to the input channel a and the transition connected with channel x are in conflict. It can never be ensured that N will always first receive an a-message whenever it is available. It may as well send an x-message which leads the net into the right hand branch where no a-message will ever be received. Thus, the a-message remains on the message channel.

The validation of N can be refined by applying behavioral constraints [2] that filter the set of partners, for instance to only those partners sending a b-message.

2.3 Operating guidelines

Though every partner of N is simulated by the most-permissive partner, the converse does not hold. To characterize exactly the set of all partners of N, we annotate the states of the most-permissive partner with Boolean formulae (see [4] for details). This annotated most-permissive partner is called an *operating guideline*. Wendy can calculate an operating guideline by generating the Boolean formulae in a post-processing step.

2.4 Reduction rules

Wendy synthesizes different types of partners depending on the analysis goal: (i) by applying no reduction rules, it synthesizes a most-permissive partner which serves as the basis for the calculation of the operating guideline; (ii) for checking controllability, the type of the synthesized partner is not of much interest. Here, we focus on a quick answer about the existence of some partner; (iii) by combining the reduction rules in a certain way, Wendy generates a particular partner which will be used later on, for instance for generating an adapter service [3].

The first three reduction rules (cf. Fig. 2(b)–(d)) focus on reducing the overhead due to asynchronous communication, whereas the last two rules (cf. Fig. 2(e)–(f)) always lead to a rather quick partner synthesis and thus an answer with respect to controllability. See [10] for further information on the reduction rules.

- *Waitstates only (WSO).* Send a message or synchronize only if it is really necessary for N to move on. Messages are not sent in advance.
- *Receiving before sending (RBS).* Before sending a message or synchronization, receive every asynchronous message sent by N.
- *Sequentialize receiving events (SRE).* Receive the messages sent by N in a certain order again. This, however, does not necessarily have to be the order the messages have been sent by N.
- *Succeeding sending event (SSE).* Quit any interaction as soon as one sent message leads to a proper interaction with N.
- *Quit as soon as possible (QSP)* Quit the interaction if any (synchronous or asynchronous) action has led the interaction with N to a proper end.

The reduction rules may be combined arbitrarily. However, only some combinations are reasonable, whereas other combinations are less appropriate. In the following, we list a few types of partners that can be synthesized by combining different reduction rules (shown in brackets at the end of each description).

- *Chatty partners* send as much as possible and receive as less as possible (SRE). These partners are best suited for adapter synthesis [3], because they hardly block the overall system, but send messages as early as possible.
- *Good Listeners* only send messages if it is really necessary (WSO, RBS).
- *Arrogant partners* let service N do all the work for proper interaction and only react if it is really necessary (WSO, RBS, SRE).

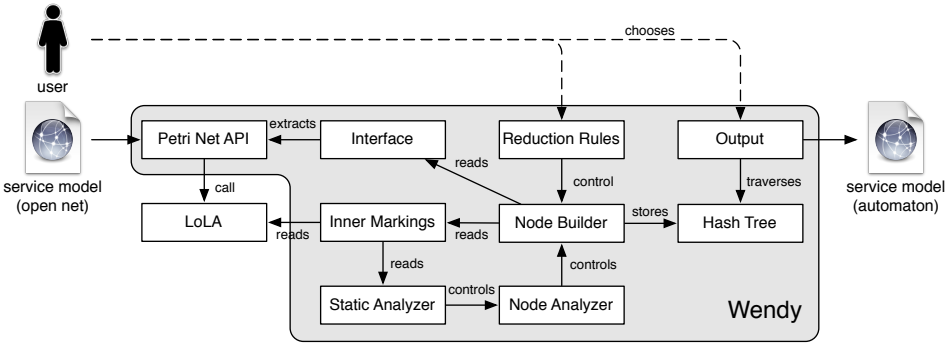


Fig. 3: Architecture overview.

- *No-talker partners (I)* only react upon the actions of service N and quit as soon as one sent message leads to a proper end (WSO, RBS, SSE).
- *No-talker partners (II)* listen, but as soon as one sent message leads to a proper end, they quit (WSO, SSE).
- *Lazy partners* do not like to interact with N (WSO, RBS, SRE, QSP).

3 Architecture

Wendy is written in C++ and built up out of several components. The overall architecture is sketched in Fig. 3. To process the input open net, given in a file format as sketched in Fig. 1(c), Wendy uses the Petri Net API[3], a C++ library encapsulating Petri net-related functions. It extracts the interface of the open net and calls LoLA [14] as an external tool to generate the state space of the *inner* of the open net (i. e., the open net without its interface).

These *inner markings* are then statically analyzed to detect deadlocks as early as possible. For instance, open net N contains the deadlock $[p^*]$ of the inner, because transition t is dead. To synthesize a partner, the node builder calculates an over-approximation (see [1] for details), and the node analyzer removes "bad" nodes. With the help of the information of the static analyzer, every node containing a deadlock — as marking $[p^*]$ in N — is declared "bad" right away, and no more successor nodes of this node are calculated. Inner markings from which a deadlock cannot be avoided anymore (e. g., marking $[p]$ in N) are treated similarly. This *early deadlock detection* has a great impact on the runtime in case an open net contains several deadlocks, for instance due to the application of behavioral constraints [2].

In addition to the node analyzer, also the chosen reduction rules influence the synthesis of the partner. The calculated nodes are compactly stored in a hash tree to quickly detect already calculated nodes. Finally, the synthesized partner is returned as either an automaton or an operating guideline.

[3] See `http://service-technology.org/pnapi`.

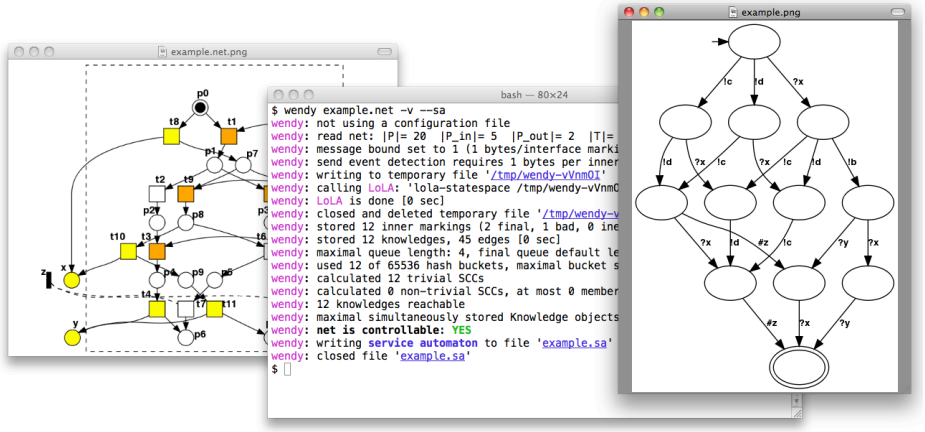


Fig. 4: A screenshot of Wendy analyzing the running example.

The core design goals of Wendy are to (1) decrease the runtime by gathering as much information about the service model during pre-processing (e. g., by analyzing the inner markings); (2) decrease the memory consumption needed to synthesize partners by implementing very problem-specific abstract data types and by keeping as little information as possible in memory.

4 Using Wendy

4.1 Use cases

Wendy is a command-line tool implementing the following use cases. We assume an open net is given as file "service.net" in the format sketched in Fig. 1(c).

- *Partner synthesis.* To check if the open net is controllable and to synthesize an unreduced partner if one is present, call Wendy with `wendy service.net --sa`.
- *Operating guidelines.* By invoking Wendy with the following command, the operating guideline of the given open net is calculated: `wendy service.net --og`
- *Reduced partner synthesis.* To synthesize a *no-talker partner (II)* by combining reduction rules *waitstates only* and *succeeding sending event*, call Wendy with `wendy service.net --sa --waitstatesOnly --succeedingSendingEvent`.

In all three use cases, Wendy will print out whether the given open net is controllable or not. If the net is controllable, Wendy generates a file "service.sa" containing the respective partner service automaton or a file "service.og" which contains the operating guideline of the service. With the command line parameter `--dot`, also a graphical representation of the output is created (cf. Fig. 4). For uncontrollable nets, the diagnosis algorithm described in [15] is implemented in Wendy. A full description of the command line parameters can be found by executing `wendy --help` or in the manual.[4]

[4] Available at `http://service-technology.org/files/wendy/wendy.pdf`.

4.2 Case studies

As a proof of concept, we synthesized unreduced and reduced partners of several WS-BPEL services from a consulting company. Each process consists of some 40 WS-BPEL activities and models communication protocols and business processes of different industrial domains. To use Wendy, we first translated the WS-BPEL processes into open nets using the compiler BPEL2oWFN [12].

Table 1 lists details on the processes as well as the experimental results for the unreduced partner synthesis. We see that the open nets derived from the WS-BPEL processes have up to 15,000 inner markings. The interfaces consist of up to 19 message channels. The number of states of the unreduced partner service (i. e., the most-permissive partner or operating guideline) are sometimes much larger than the original service. The number of transitions grows even faster. The analysis takes up to 300 seconds on a 3 GHz computer. Given the fact that operating guidelines are usually calculated only once and are to be used by the service broker many times, this is satisfactory. It is noticeable that the calculation of the operating guideline's formulae took only a split of a second for all models.

To further evaluate Wendy, we processed some parametrized academic benchmarks within a 2 GB memory limit. Figure 5 illustrates that Wendy is capable of analyzing service models with up to 5,000,000 inner markings and to synthesize partner services with up to 4,000,000 states. At the same time, we see that the largest industrial models we analyzed (14,990 inner markings and 57,996 partner states) do not come close to these bounds.

The effect of the reduction rules is summarized in Tab. 2. Depending on the applied reduction rule, the analysis time can be dramatically reduced. Using the rule *quit as soon as possible*, for instance, allows to calculate a partner service in at most 2 seconds for all services. This result can be generalized to any other open net we analyzed so far: if the inner is bounded (controllability is undecidable if the inner is unbounded [17]), Wendy could always decide controllability by applying reduction rules.

4.3 Comparison with other tools

Tools from classical controller synthesis [18] are not really comparable to Wendy, because they consider only synchronous communication (whereas Wendy also supports asynchronous communication which is crucial in SOC) and make different assumptions on the observability of internal states and events.

Both the partner synthesis algorithm and the algorithm to calculate an operating guideline for a service have been previously implemented in the tool Fiona [19]. The design goal of Fiona was the combination of several analysis and synthesis algorithms for service behavior. This is reflected by a flexible architecture that aims at the reusability of data structures and algorithms. Though this design facilitated the quick integration and validation of new algorithms, the growing complexity made optimizations more and more complicated. To overcome these efficiency problems, Wendy is a reimplementation of the synthesis algorithms as a compact single-purpose tool. This reimplementation incorporates

Table 1: Unreduced partner synthesis for industrial WS-BPEL services.

	analyzed service			synthesized partner		
service	inner markings	transitions	interface	states	transitions	time
Quotation	602	1,141	19	11,264	145,811	0
Deliver goods	4,148	13,832	14	1,376	13,838	2
SMTP protocol	8,345	34,941	12	20,818	144,940	29
Car analysis	11,381	39,865	15	1,448	13,863	49
Identity card	14,569	71,332	11	1,536	15,115	82
Product order	14,990	50,193	16	57,996	691,414	294

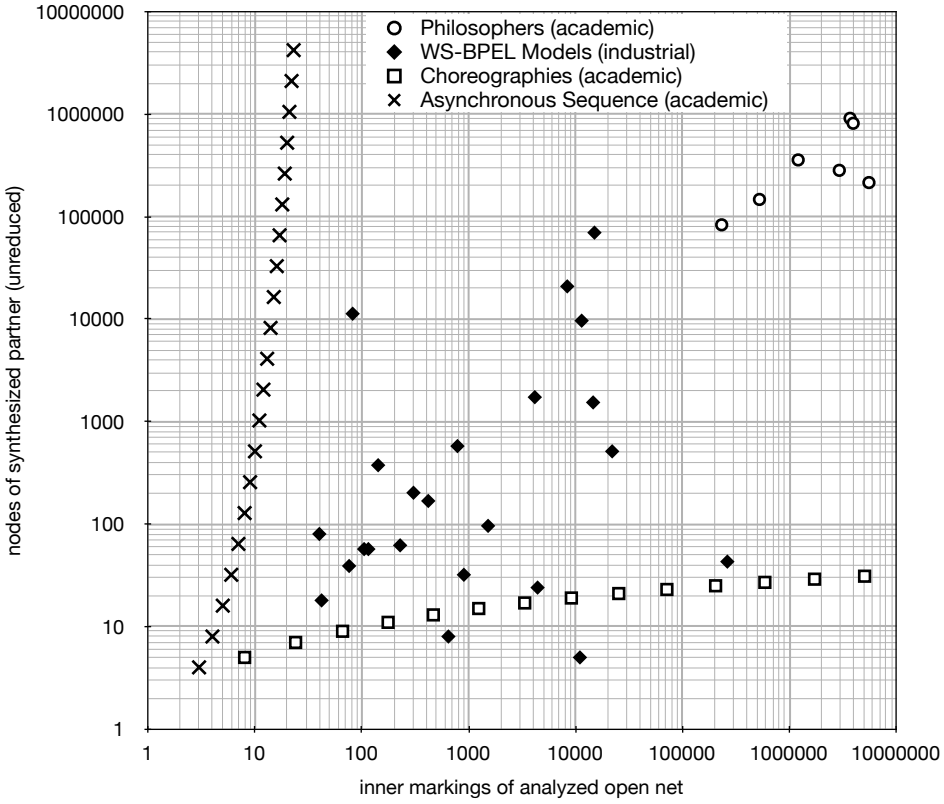


Fig. 5: Limits of the synthesis algorithm (2 GB of RAM). Beside some industrial WS-BPEL models (♦), we processed several parametrized academic benchmarks: *Asynchronous sequence* (×) is a family of services with exponential growth of states of the partner services; *Choreographies* (□) are BPEL4Chor choreographies from [16] with an exponential growth of inner markings; *Philosophers* (○) is a benchmark set of the WODES workshop, see `http://www.wodes2008.org/pages/benchmark.php`.

Table 2: Partner synthesis using reduction rules.

	WSO		RBS		SRE		SSE		QSP	
service	states	t	states	t	states	t	states	t	states	t
Quotation	6,244	0	1,667	0	2,287	0	52	0	20	0
Deliver goods	1,328	2	89	0	154	0	65	0	16	0
SMTP protocol	3,270	11	4,872	0	16,837	18	30	0	30	0
Car analysis	1,448	47	108	1	96	11	78	28	38	2
Identity card	1,280	74	261	1	48	2	1,280	74	12	0
Product order	57,762	300	741	1	771	3	1,782	9	101	0

the experiments made by analyzing performance bottlenecks through improved data structures and memory management, validation of case studies which gave a better understanding of the parameters of the models that affect scalability, and theoretical observations on regularities of synthesized strategies and operating guidelines.

In comparison, Fiona could only analyze three out of the six services presented in Table 1 without taking more than 2 GB of memory. For the other services, the analysis was between 5 and 70 times slower than Wendy. In addition to Fiona, Wendy handles synchronous communication and implements two more reduction rules (*succeeding sending event* and *quit as soon as possible*).

4.4 Obtain Wendy

Wendy is free software and is licensed under the GNU AGPL.[5] The most recent version is available from Wendy's website at `http://service-technology.org/wendy`. There, the source code as well as precompiled binaries can be downloaded. To compile Wendy from its sources, only a recent version of the GNU Compiler Collection (gcc) is required. We tested several platforms including Windows, Mac OS, Linux, FreeBSD, and Solaris. In addition, a demo version is accessible at `http://service-technology.org/live/wendy` where the examples of this paper can be replayed in a Web browser.

5 Conclusion

The functionality provided by Wendy — the synthesis of partners for services — is a basis of a variety of important applications in the paradigm of SOC. To this end, Wendy is already integrated into tools realizing adapter synthesis [3] and instance migration [7]. Case studies show that Wendy can be used in academic as well as industrial settings.

In future work, we plan to adjust the synthesis algorithm to exploit the multiple cores that are increasingly present in personal computers. In addition, symbolic representations such as BDDs [20] may further help to reduce the memory consumption during the synthesis. At the same time, reduction techniques already

[5] GNU Affero Public License Version 3, `http://www.gnu.org/licenses/agpl.html`.

implemented in LoLA might be applicable when calculating the inner markings of the given open net.

Acknowledgments. The authors thank Stephan Mennicke and Christian Sura for their work on the Petri Net API as well as Karsten Wolf for sharing experience made with LoLA and valuable discussions on the data structures.

References

1. Wolf, K.: Does my service have partners? LNCS ToPNoC **5460**(II) (2009) 152–171
2. Lohmann, N., Massuthe, P., Wolf, K.: Behavioral constraints for services. In: BPM 2007. LNCS 4714, Springer-Verlag (2007) 271–287
3. 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 (2008)
4. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: PETRI NETS 2007. LNCS 4546, Springer (2007) 321–341
5. Kaschner, K., Lohmann, N.: Automatic test case generation for interacting services. In: ICSOC 2008 Workshops. LNCS 5472, Springer (2009) 66–78
6. Lohmann, N.: Correcting deadlocking service choreographies using a simulation-based graph edit distance. In: BPM 2008. LNCS 5240, Springer-Verlag (2008) 132–147
7. Liske, N., Lohmann, N., Stahl, C., Wolf, K.: Another approach to service instance migration. In: ICSOC 2009. LNCS 5900, Springer (2009) 607–621
8. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding substitutability of services with operating guidelines. LNCS ToPNoC **II**(5460) (2009) 172–191
9. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WS-BPEL processes using flexible model generation. Data Knowl. Eng. **64**(1) (2008) 38–54
10. Weinberg, D.: Efficient controllability analysis of open nets. In: WS-FM 2008. LNCS 5387, Springer (2009) 224–239
11. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA. Annals of Mathematics, Computing & Teleinformatics **1**(3) (2005) 35–43
12. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: WS-FM 2007. LNCS 4937, Springer (2008) 77–91
13. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers. Trans. Inf. and Syst. **E80-D**(3) (March 1997) 315–325
14. Wolf, K.: Generating Petri net state spaces. In: PETRI NETS 2007. LNCS 4546, Springer (2007) 29–42 Invited lecture.
15. Lohmann, N.: Why does my service have no partners? In: WS-FM 2008. LNCS 5387, Springer-Verlag (2009) 191–206
16. Lohmann, N., Kopp, O., Leymann, F., Reisig, W.: Analyzing BPEL4Chor: Verification and participant synthesis. In: WS-FM 2007. LNCS 4937, Springer (2008) 46–60
17. Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Undecidablity of partner existence for open nets. Inf. Process. Lett. **108**(6) (2008) 374–378
18. Ramadge, P., Wonham, W.: The control of discrete-event systems. Proceedings of the IEEE **77**(1) (1989) 81–98
19. Massuthe, P., Weinberg, D.: Fiona: A tool to analyze interacting open nets. In: AWPN 2008. CEUR Workshop Proceedings Vol. 380, CEUR-WS.org (2008) 99–104
20. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. Computers **C-35**(8) (1986) 677–691