

Operating Guidelines for Service Oriented Architectures

Wolfgang Reisig

joint work with
 Peter Massuthe and Karsten Schmidt

1

1. what is this talk about?

the background

Service = id + control + interface

Web Service: id = URI, interface = WSDL,

Workflow Service: control = workflow

Workflow = implemented business process

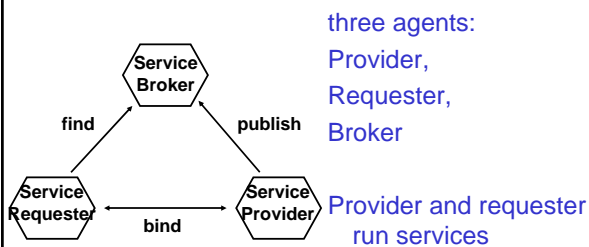
examples:

online banking car rental Java program

$WS \wedge WFS$ $\neg WS \wedge WFS$ $WS \wedge \neg WFS$ ²

Service-Oriented Architecture

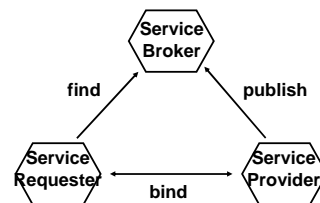
SOA – framework for Service interaction



3

The problem

a requester seeks a provider or
 a provider seeks a requester
 for joint execution of their respective service



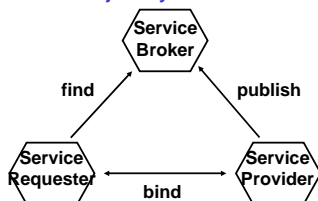
4

The solution

publish: Provider sends information to a broker on how to interact with his service + his id

find: Requester sends his service to a broker; Broker selects fitting service and returns Provider id

bind: Requester establishes connection with a provider, and both jointly run their services.



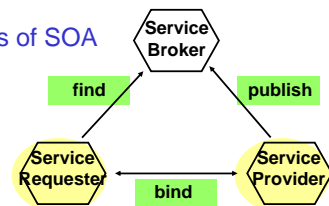
5

Topic of this Talk

formal models of

- Workflow Services,
 in particular services of providers and requesters

- the three tasks of SOA



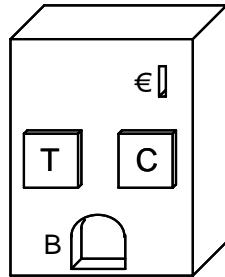
6

2. How model services?

example:

a vending machine

- expects a coin be inserted
- expects a button T (tea) or C (coffee) be pressed
- delivers the beverage



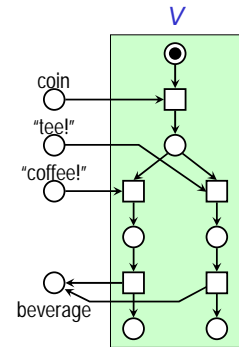
7

Model of the Vending Machine

example:

a vending machine

- expects a coin be inserted
- expects a button T (tea) or C (coffee) be pressed
- delivers the beverage



8

Model of the Vending Machine

Petri net V , with

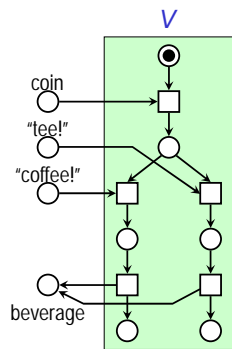
an *initial marking*
two *final markings*

input places

coin, "tee!" "coffee!"

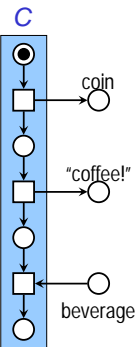
an *output place*, beverage

service of a provider



9

Model of a coffee customer



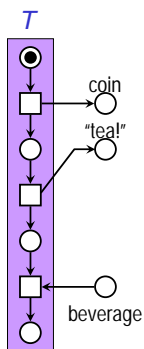
- inserts a coin
- presses the coffee button
- receives the beverage

again a Petri Net, with

output places, coin, "coffee!"
an input place, beverage

10

Model of a tea customer



- inserts a coin
- presses the tea button
- receives the beverage

again a Petri Net, with
output places, coin, "tee!"
an input place, beverage

11

Petri Net Models of Services

open workflow nets

a liberal version of *v.d.Aalst WF-Nets*
enriched with input/output places

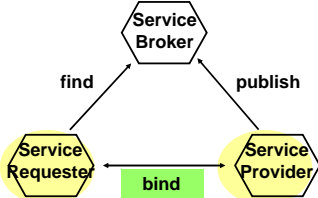
12

3. how model *bind* ?

remember: topics of this talk: formal models of

- Services, in particular services of providers and requesters
- the three tasks of SOA

done!

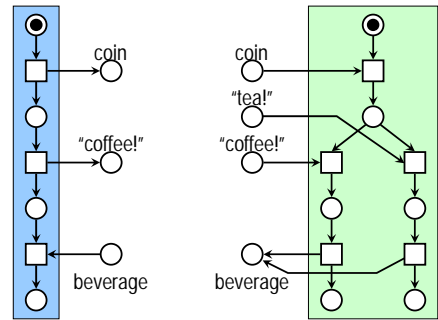


next problem:

how model *bind*

13

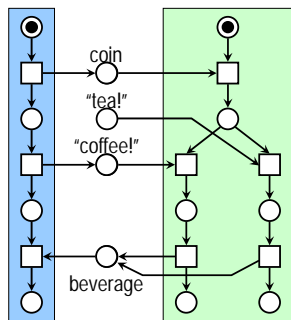
bind is composition of oWFNs glue shared input/output places



14

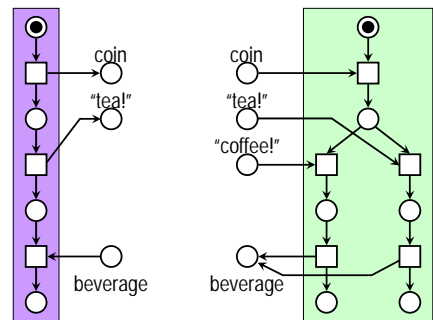
bind is composition of oWFNs glue shared input/output places

the oWFNs jointly reach a terminal state



15

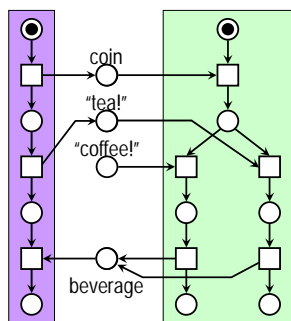
bind is composition of oWFNs glue shared input/output places



16

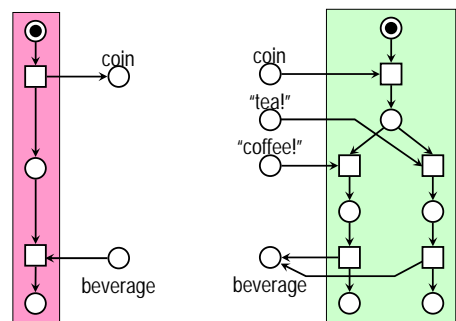
bind is composition of oWFNs glue shared input/output places

the oWFNs jointly reach a terminal state



17

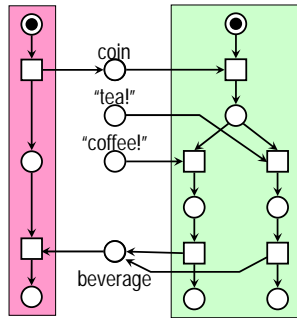
an inadequate customer



18

an inadequate customer

the oWFNs jointly get stuck



19

Composition of oWFNs R and P

Identify common interface places.
Yields $R \oplus P$.

Observation. $R \oplus P = P \oplus R$.

20

Strategy: an adequately cooperating service

Def. Let R and P be oWFNs.

R is a *strategy* for P , iff every deadlock in $R \oplus P$ is a final marking

above examples:

C and T are strategies for V

The *inadequate customer* is no strategy for V

21

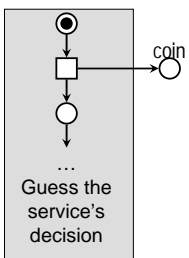
problems concerning strategies

- Let P be an oWFN.
- Is a given oWFN R a strategy for P ?
- does there exist a strategy for P ?
- how construct a strategy for P ?

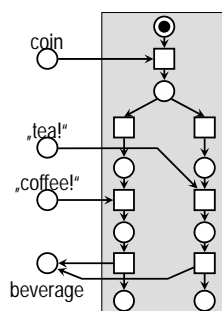
! there exist efficient algorithms for those problems!

22

Variant: *Kafka's* vending machine

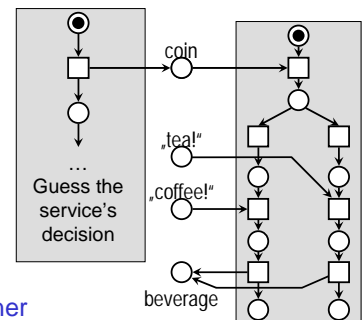


... Kafka has no partner



23

Variant: *Kafka's* vending machine



... Kafka has no partner

24

4. How model *publish* ?

more precisely:

What should the provider publish?

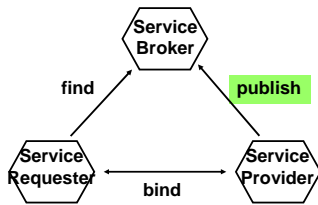
frequently suggested: *public view*

Problems:

In general there is no canonical candidate.

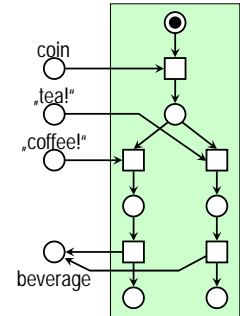
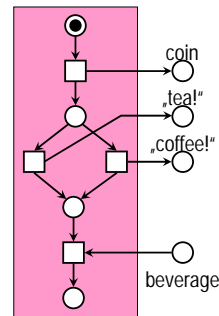
Public view is not what a requester wants.

We suggest instead:
"operating guidelines"



25

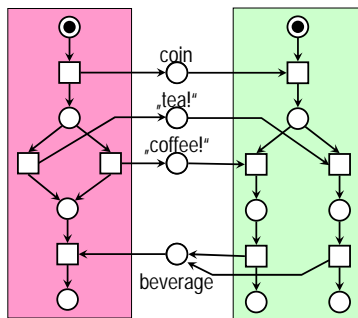
A More Permissive Strategy for V



26

A More Permissive Strategy for V

terminates perfectly

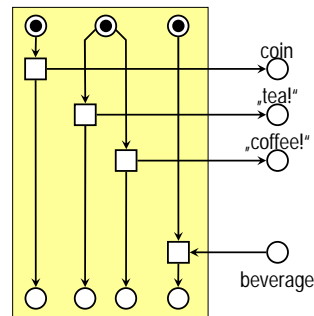


27

The most permissive strategy

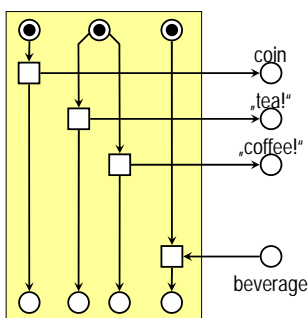
idea:

"inserts a coin"
"presses a button"
may be executed concurrently.



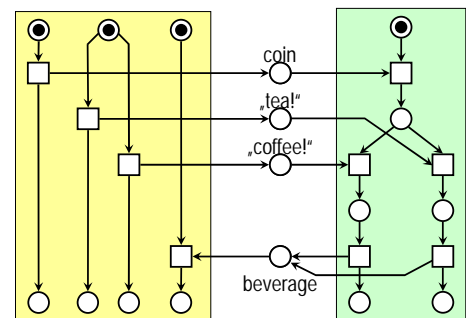
28

The most permissive strategy



29

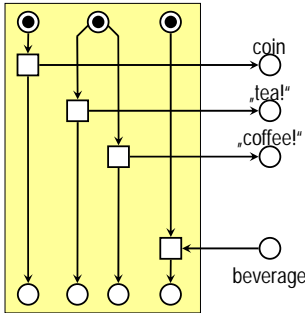
The most permissive strategy



30

Operating Guidelines for P:

set of *all* strategies for P.



technically:
the most permissive
strategy + some
minor knowledge on P.

publish =_{def}
Operating Guidelines

31

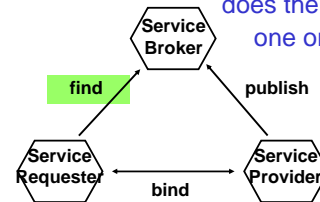
4. How model *find* ?

given:

- Broker, managing the operating guidelines for each Service Provider
- a Requester, searching a fitting Provider

the technical problem:

does the Requester's service match
one or more operating guidelines
managed by the Broker?



can algorithmically
be decided

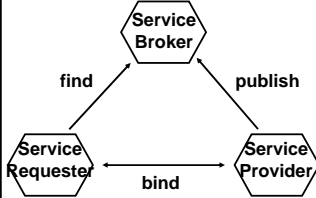
32

What did we achieve?

a formal model for all essential components of SOA,
including fundamental notion for
Workflow Service (control = workflow)

technically:

open WF-Nets (oWfN) with *composition, strategy,*
Operating Guidelines



33

How continue ?

case studies

versions of strategies

ports

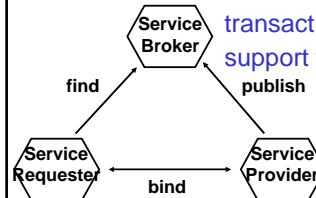
more efficient analysis techniques

abstraction/refinement

simulation, equivalence

transactions

support for languages such as BPEL



34

The End.

many thanks

35

Operating Guidelines: An Alternative to Public View P. Massuthe, W. Reisig, K. Schmidt

Service Oriented Architecture (SOA) is a promising and influential software architecture. Its basics include distributed, communicating *services*. As a typical structure, a *service provider* offers information to *service requestors* on a service's capabilities. The service provider thereby intends the amount of information to remain at a bare minimum, mainly to cover business secrets, but also to shield the requestor from unnecessary details. To this end, it was suggested to publish a service *S* together with a *public view*, i.e. an abstract version of *S*, sufficient to derive information to properly communicate with *S*. Formal definitions of public views are still missing, and some case studies in fact reveal more than one reasonable candidate for the public view of a service.

As an alternative to public views, we suggest *operating guidelines*. An operating guideline describes the adequate behaviour of the user of a service directly.

As an example, let *M* be a vending machine. The public view of *M* would be some kind of abstract description of how *M* actually works, from which the customer is expected to figure out at which time he or she is expected to insert a coin, or to push a button. Such a description is not what we usually find pinned to a vending machine. Instead, we usually find (if anything) operating guidelines, i.e. an algorithm that is expected to be run by the user (possibly including choices that are resolved depending on the user's intention). That is, rather than a description of *M*'s behaviour, there are operating guidelines, describing the expected *customer's* behaviour.

We show that it is in fact possible to automatically generate such operating guidelines from a business process model. In many situations, it is even possible to achieve completeness of these guidelines. Completeness means that *every* behaviour of a communication partner that *violates* the operating guidelines may cause deadlocks, or erroneous messages.

36