Preprint

This is the submitted version of a paper presented at AWPN 2012: Algorithmen und Werkzeuge für Petrinetze.

Citation for the original published paper:

Jalali, A. (2012)
Service Oriented Modularization using Coloured Petri Nets.
In:

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-100994
Abstract. Modelling service oriented systems using Coloured Petri Nets usually results in cluttered nets which are hard to understand and modify. This complexity is a result of many interactions among services. This paper presents a method for designing service oriented models using coloured petri nets. This method results in less complex nets which could be extended easier. The validation of the method is given through demonstrating its impact on defining operational semantics of a service.

Keywords: Coloured Petri Nets, Service Oriented, Modularization, Message Oriented

1 Introduction

Coloured Petri Nets (CPNs) have been widely used for the design, specification, simulation, validation and implementation of systems [5]. A type of systems is service oriented system which consist several services communicating each other. This communication is performed using an interface. The interface is modelled in petri nets using a place [7]. Moreover, some services return result when they are invoked, so two places could be used in modelling such services to model request and response interfaces correspondingly [2].

However, if there are a lot of services in a system, the result will be a very complex and un-organized model. Figure 1(a) represents this problem in programming area where services invoke each other directly using Remote Procedure Call (RPC). This problem is addressed by decoupling services using a message broker, called Message Oriented Middleware (MOM). Figure 1(b) represents communications between a number of services using MOM. This middleware adds the level of abstraction in models in a way that services do not require to know each other addresses. Each service sends and receives messages through the broker, so the model will be simpler by utilizing a unique communication channel [6], and the maintenance and development of model will be easier.

This means of communication could not be utilized in petri nets, because it allows any sequence of occurrence between services to happen. Such a petri net is called flower model [1]. On the contrary, this pattern could be followed by CPN models, because each token has a colour which could be distinguished. Thus, the designer could define the occurrence sequence of services in CPN, which will be explained in this paper.
In this paper, we propose a method to design service oriented systems using CPN. This method enables service oriented modularization and separation of concerns that results in less complex nets which could be modified and maintained easier.

2 Method

Our method follows the top-down service definition approach. The following steps describe how service oriented systems could be modelled using this method:

1. Recognize services which collaborate in the system.
2. Define a hierarchical module for each service.
3. Connect all services together using a common place, called broker. This broker will be used by services as an input and output place.
4. Define the colourset for the broker in a way to consist of command and data, so all services can send and receive appropriate messages to and from the broker.
5. Define the semantics of each service in each module separately.

3 Application

In this section, we introduce how to apply this method using a simple example. Figure 2 shows a net capturing an agent which aims to calculate some statements by utilizing different calculator services. The net consists of four services, i.e. Agent, Interface Area, Simple Calculator Service and Advanced Calculator Service. The Agent produces tokens in some places in first, oper and second places in Interface Area which are used as input places. The Agent receives the result as a token in result place in Interface Area which is used as output place.

The Interface Area consists of input and output places which decoupled services from each other. The calculators consumes tokens from input places upon satisfying the guard condition(contract). They return the result to the output place. As it could be seen, the nets could be very messy by adding more...
services or adding functionality to each service. For example, if we want to multiple and divide functions to the Simple Calculator Service, this service will be very messy. Therefore, we apply our method to end up with better encapsulation and modularization.

Figure 3 shows the result when applying our method. The services are encapsulated in three modules, which are connected to each other using common place, Broker. This place has a MSG colourset. This colourset consists of service name (called command) and data. The service name is the operator that specifies the service and related function to be invoked. The data are the first and second operands. In this way, each service is encapsulated in a module, and each module communicates to the broker for a request or response.

4 Discussion

In this section, we discuss this modeling technique by comparing two nets.

– The first nets have more arcs and places for interface area, this makes the model complex. This complexity makes it difficult to extend and maintain the nets. The second nets hide this complexity using a common place. This place is used as a single channel of communication between services, so the number of places is reduced. Moreover, the arcs which connect transitions to the common place are encapsulated in services, so they are more understandable and organized. Therefore, it is easier to maintain and develop the nets using this technique, and the nets are more readable.

– Theses nets calculate the input statements in similar sequence, so their The examination of all markings and state transitions in these examples indicates that the two state spaces capture the same information though it is captured
5 Validation

This method is tested through a case study in which we defined the operational semantics of a service which extends the Business Process Management Systems to support Aspect Orientation [4, 3]. Figure 5 shows two nets. The above net shows the usual modelling technique to define the semantics for a service. This service is composed of several services, so the nets are very messy. In addition, it was cumbersome to change the model or add new functionalities to it due to its complexity.

The bottom part of the figure shows another net which is the result of applying this method. The nets are decomposed based on different services which collaborate using a common place, called ICore. This place has a colourset, COREMSG, that is defined as a production of CMD and PARMS. CMD is defined as

\[ CMD = \text{product} \times \text{STRING} \times \text{STRING} \]

PARMS is defined as

\[ PARMS = \text{product} \times \text{STRING} \times \text{STRING} \]

1 The nets are available for download in www.aobpm.com
specific command which determines which service should be invoked. **PARAMS** defines the data. This colourset is defined as a list of **PARAM**. **PARAM** is a production of two strings that represent the name and value of the parameter. In this way, each token could represent not only the command, but also infinite number of arguments which are customized for each service.

Furthermore, this modelling technique enables services to call each other in a chain. For example, it is possible that service A calls service B, and service B call service C, and service C returns the result back to service A. This provides a very flexible design which facilitates separation of concerns and encapsulation.

### 6 Conclusion

In this paper, we presented a new method to modularize Service Oriented Systems using Coloured Petri Nets. This method is based on the Message Oriented Middleware idea which proposes to use a single layer as communication channel between services. The method is validated using a case study. In the case study, we defined the operational semantics of a service to extend the behaviour of Business Process Management Systems to support Aspect Orientation, a means to support separation of concerns. This case study reveals how this method results in more flexible nets with better encapsulation and separation of concerns.

This method is very useful in designing new nets; however, it requires more research on how different designed nets could be merged and used in a new
system. In programming, this problem is addressed using Enterprise Service Bus (ESB) which could also be investigated as future research.

Acknowledgement
I would like to thank Chun Ouyang and Petia Wohed for their valuable comments and support during writing this paper.

References