Service Modelling and Verification: A Formal Approach

Deepak Chenthati and Hrushikesha Mohanty

Abstract Increasing number of users on web has attracted a large number of businesses to be made available on web. The growing demand for webservices requires a systematic approach in service development. For the purpose, this chapter reviews various models for service specification, composition, deployment and monitoring. Particularly, the chapter reviews some well-known models like UML, Petrinet and state machine used for service modelling and verification.

1 Introduction

Web has become a common platform for different enterprises with increasing availability of the Internet. Also with standardisation of SOA technologies, enterprises tend to expose their business as webservices. Enterprises to meet the market needs would resort to the methods that enable them to build a service quickly and effectively without errors. In this process, either the existing services are used for building a service with enhanced/higher requirement or services are generated from models. Here, initially, the services are modelled using formal methods, viz. UML, Petrinet, state machine, etc. A modelled service is used as a source/reference for automatic generation of executable codes. The advantage of service modelling is in three folds. Firstly, it gives a visual representation of a service that is being modelled giving service provider a scope to analyse its flow of execution. Secondly, the errors due to manual coding are eliminated. Thirdly, the deployed/running service can be verified

D. Chenthati (✉)
Teradata (R&D) India Pvt Ltd, Hyderabad, India
e-mail: chvcdeepak@gmail.com

H. Mohanty
Kalinga Institute of Industrial Technology, Deemed to be University, Bhubaneswar 751024, India
e-mail: hmcs_hcu@yahoo.com

H. Mohanty
School of Computer & Information Sciences (On leave), University of Hyderabad, Hyderabad, India

H. Mohanty and P. K. Pattnaik (eds.), *Webservices*,
https://doi.org/10.1007/978-981-13-3224-1_1

against the model either for testing or for checking the abnormal behaviour of a service. These advantages have motivated several research works to address modelling of services considering different aspects.

A webservice has a business logic available on web and it is exposed to the external world as an interface. The major concerns during modelling of services would be (i) to hide the business logic, (ii) to generate an interface as per a specification standard, i.e. WSDL, (iii) composing a service when a single service does not meet a need, (iv) monitoring a deployed service to capture quality details, (v) security aspects of a service and (vi) provision for model-based verification. Each model follows different rules for modelling and code generation considering some or all of the aspects stated above. These rules in turn form the guiding principles for verification of webservices.

The concern of business logic is addressed with orchestration of a service where the activities of a service are in order for execution. Composition of services involves issues with matching of interfaces and protocol for proper communication among constituting services. Matching of I/O interfaces would initially find a set of services that are further checked for protocol match. Choreography of webservices addresses this issue assuring observance of an agreed protocol among chosen services. Further, the verification of a service specified with rigour of models addresses issues related to structural and behavioural correctness. However, here no claim is made on completeness of model-based webservice specification.

Section 2 of this chapter gives a detailed review of some models used for service modelling. The next section takes up issues on verification of services. Modelling of services is discussed with respect to approaches based on UML, Petrinet and state machine. Each of these approaches is discussed with different enhancements and proposed standards. Finally, the chapter concludes with a brief concluding remark.

2 Modelling of Webservices

2.1 UML-Based Approach

Unified modelling language (UML) has been successful in modelling software systems that is reflected from its popularity in software industry. Webservices being a contemporary technology-driven means to deliver services, needs well-thought design principles for success. UML being a proven modelling approach is considered for the purpose of webservice design. This, in fact, brings both academia as well as industry professionals to the same platform to contribute together in development of UML-based methodology for webservice development. In this section, we will survey some works in highlighting UML-based webservice design approach.

Model-driven architecture (MDA) is a popular method to develop software systems. UML has a mechanism to model both structural as well as behavioural specifications of a system. In case of webservices, UDDI that maintains service locations also comes to picture. Further, composition of services is to be modelled. Service compo-

sition modelling specifies a way messages between two services are to be exchanged. Compatibility of message passing protocol specification leads to successful service composition. UML class diagram models interface while actions are modelled by UML behavioural diagrams like use case, activity and interaction diagrams. The aspects particularly to be modelled with respect to webservices include service artefacts, interfaces, data access, execution and communication error handling, execution tracing, usages of ontology, designing service communication patterns, specifying service locations and service metamodelling. A brief survey on usability of UML in specifying webservice composition is given in [1].

W3C, the organisation engaged in developing standard has proposed a specification web services choreography description language (WS-CDL) for specification of webservices. Service providers specify services and for a consumer request, services are searched and an appropriate service is selected for providing service. Researcher in [2] proposes a variant of UML-2.0 called UML-S to specify a system to be developed with WS-CDL specification. This requires a one-to-one mapping between the two specification approaches, i.e. WS-CDL and UML-S. Service modules specified in WS-CDL are specified in UML component diagrams. A sequence of actions a module performs is specified by UML activity diagrams. Behaviours due to each role specified in WS-CDL is modelled by UML state diagrams, whereas descriptions for each state is modelled by class diagrams. With an experimentation, authors have shown the use of UML-S-based specification in modelling and verifying a service specification.

In [3], researchers have proposed another extension to UML called WS-UML for webservice design. WS-UML is alike UML, proposes graphical annotations for specifying webservice concepts like service security, composition, location and trace of execution. This work stands out among similar works by allowing to specify service location. It also allows user service providers as well as consumers to trace service execution. A composed webservice is to provide an integrated service to a user in accordance with its requirements. This requires selection of services and composition of services. This process requires certain essential aspects that a service designer must address. WS-UML provides that mechanism for specification.

First, question of service selection comes. Selection of a service not only is based on service functionality say by specifying input and output but also its quality of services. Quality of a service for a webservice is specified by service cost, reliability and execution time required for service delivery. WS-UML provides means to annotate a service by these information in a given syntax. Further, on selecting services, compositions are done. In the process, first composability criteria are to be evaluated. Each service specifies its conditionality for composition, e.g. protocol requirements for passing messages. Other than this communication, there is a need to specify security aspect. Designer makes a provision for defining security mechanism for both service provider as well as consumer. Hence, during invocation of service security aspects for both provider and consumer can be ascertained. Location where a service is hosted on the Internet is to be specified for easy access of users. This access is to be of course automated. For the purpose location specification is necessary. This provision is made in WS-UML. Next is execution tracing that is essential in business

world. Once a service is invoked, it is required to trace the execution to check whether the service provision is being done as per requirement, i.e. service-level agreement.

WS-UML provides means to specify the desired states by which service execution must pass through. This design provision helps in service error finding and thus service maintenance.

Later a work [4] proposes a process of webservice design using UML. Webservice is seen as a choreography of several loosely coupled services. They collaborate to solve a common task regardless of their programming languages and environment. In order to model such collaborative system, UML is considered as a natural choice. Because, component diagram, sequence diagram and state diagram can model service components, their functionalities, interactions and state changes during execution. The paper proposes seven steps for webservice choreography modelling and verification. First, WS-CDL structural properties are modelled by component diagrams. In the second step, sequence diagrams used to model component interactions are translated to state diagrams. Because, state diagrams can be used for model verification. Then, in the third step, the abstract data model for WSDL is modelled by class diagrams. State diagrams are now enriched by class diagrams with tags stating the changes occurring to data in different states. Then, in the fifth step, enriched state machine diagram is translated into language of a model checker like SMV. Then, in the following sixth step, informal requirements, if any, are specified by designers for verification. At the last step, a model checker is used to verify a model behaviour. This work shows not only how to model a webservice but also to verify its design specification.

In another work [5], use of RT-UML to model and verify orchestration of webservices is reported. The main purpose is to verify time aspects of orchestration. For this purpose, RT-UML is considered. Researchers have found one-to-one mapping between RT-UML constructs and WS-BPEL. A top-down design approach is proposed for developing webservices. During analysis, time constraints that webservice orchestration must satisfy are found. And these time constraints are annotated to sequence diagram. Then, model checking approach is used to discover specification errors. This is done by translating RT-UML diagrams into timed automata that are used to perform model checking. Verified RT-UML design diagrams are then translated to WS-BPEL. The elements used by RT-UML for webservice design includes RTDelay, RTEvent, RTAction, RTreset and RTclock. The elements have their semantics that are well understood by their names. These elements have corresponding translations to WS-BPEL language. Thus, the work provides a systematic approach in design of webservices using RT-UML particularly keeping time constraints in view. The work is in importance for bringing time aspects in modelling service orchestration.

Recent work focusses beyond UML while modelling webservices. This is so for variety of emerging distributed architectures like cloud and mobile systems. The argument on utility of UML is raised for heterogeneity these platforms bring in. It cites the inability of UML diagrams in representing all the heterogeneities as inherently class diagrams model homogeneity of design entities. As a solution to this problem, researchers in [6] have proposed domain specific language (DSL) to design

webservices that runs in different environments. Design abstraction that is common to all the implementations is extracted and then the abstraction extended for each implementation. This design extension is carried out to meet the requirement of a platform. The researchers have proposed simple web service modelling (SWSM) domain specific language. Webservices in multitenant platforms are specified in SWSM language. Then, it is customised for a specific tenant. The challenge in having such a language is to specify webservice architecture at an abstract level separating logic from its technical implementation aspects.

SWSM language proposes a syntax that elegantly specifies an abstraction of webservices. The terms used in syntax are *Webservice*, *Port*, *Binding*, *Datatype*, *Operation*, *Message* and *OperationBinding*. Modelling of webservices are carried forward representing principal elements using these terms. The design approach is essentially top-down. The steps in developing webservices include modelling using SWSM language, enhancement and automatic validation of webservice models, code generation and code refinement, and refactoring and testing. Model-driven development of software systems is successful and UML has given impetus for it. But, non-UML-based modelling approach is also followed for its conceptual clarity and simplicity. This work is of that kind introducing non-UML-based webservice modelling. Next sections of the chapter will review some of these well-known modelling schemes based on finite state machine (FSM) and Petrinet.

Table 1 gives a summary on research works mapping webservice specifications to concepts in UML and this table is cited from research work in [3]. WSDL is generated from UML profile [7], an extension to UML class diagram with service-related concepts, viz. Service, Port, Port Type. Jeng and Tsai [8] look services as components and proposed UML profile based on service component architecture (SCA). Jeng and Tsai [8] views from security angle and has proposed extra functional properties to record service invocation logs, control access to operations and encryption. Composition of services is focused on [9].

2.2 Petrinet-Based Approach

Webservice while taking business on the Internet to a reality there has been spurt in research activities in this field. Service modelling and composition are the two important issues under consideration, while modelling concentrates on both static as well as dynamic aspects of webservices there has been further interest in modelling temporal as well as asynchrony aspects of webservice execution. For the purpose, Petrinet is a chosen model. Many have tried with variant of Petrinet models and shown their utilities in webservice modelling, code generation and more importantly in design verification. Here, we will review few papers that only represent types of work researchers engaged in this field. We do not claim the review is complete or exemplary. For the purpose, here, we believe the papers selected here for review are representative.

Table 1 Review of webservice specifications in UML

View	Concept	References
WSDL	Class UML stereotyped "WebService"	[7, 10]
	Class UML stereotyped "Port"	
	Class UML stereotyped "PortType"	
	Tagged value relative to a service and the port {URI="/ "}	
SCA	Component stereotyped "ServiceComponent"	[11]
	Tagged value relative to an interface {R.uri= "Operationuri"}	
	Class stereotyped "ServiceInterface"	
Security	Note stereotyped "Login"	[11]
	Note stereotyped "Log"	
	Note stereotyped "Encryption"	
Composition and orchestration	Class stereotyped "ServiceWebAtomic"	[9, 10, 12]
	Class stereotyped "ServiceWebComposite"	
	Class stereotyped "ModeledOrchestration"	
	Activity stereotyped "ImmediateStep"	
	Note stereotyped "DataTransformation"	
	Activity stereotyped "DataMapper"	
	Tagged value relative to the activity "ImmediateStep" {DomainObject=}	
QoS	Class stereotyped "ServiceQuality"	[10, 12]
Community and function	Class stereotyped "Community" Class stereotyped "Function"	[10, 12]

A webservice at higher level can be viewed as a collection of interacting modules. Architecture of a webservice is built with these modules. Configuration of modules, the way these are connected, presents an architecture for a webservice. The paper [13] presents WS-Net, an architectural description language for specifying webservice architecture. The language is based on Petrinet semantics accompanied with object-oriented paradigm. These two aspects of WS-Net provide a mechanism to specify and verify a webservice model. It is also useful to monitor dynamic behaviour of webservices. WS-Net uses coloured Petrinet for higher level design.

WS-Net is executable architectural language. Webservice model is seen here as three-layered model consisting of interface net, interconnection net and interoperation net. Interface net models network of interfaces and their possible transitions. Each service component is treated as a place and the modules that can be invoked from the place are connected by transitions. Thus, Petrinet becomes an obvious choice for webservice modelling. The next layer below, interconnection net models the foreign transitions that invoke the components that are external to a component. The next lower level models behaviour of a component, that is, a sequence of invocations of the units that make a component. Coloured Petrinet is used for understanding like to distinguish models at different levels. It is shown that the executable architectural language becomes useful for both modelling as well as verification. Adopting object-oriented paradigm in WS-Net, understandability as well as agility in webservice design are achieved.

A new concept in modelling and analysing webservice composition using Petrinet is proposed in [14]. This paper reviews all existing methods and proposes a method that illustrates modelling issues at different levels using coloured Petrinet. Basic questions like *who* and *why* for service composition are answered in this paper. Among existing set of services which one is to be selected for composition and how does the selected one can interact with other services for a given user requirement are answered by the model. The unique issue the paper takes up is selection of a service that is to be guided not only by functional requirements but also nonfunctional requirements. The cited work extends formalism of CPN to model transaction-driven composition process. Further, while modelling quality of service (QoS) is also taken into consideration. The composition process also considers system modularity. The process is standardised to make it compatible with Web 3.0. This modelling helps service composition operationally simple. A service user just puts a query stating its requirements and then the composition process matches the Petrinet-based modelled webservices and finds the services that meet both functional and nonfunctional requirements.

The paper [15] has proposed a high-level Petrinet for service modelling. The modelling concept is based on G-Net [16]. G-Net provides an algebraic specification to specify modular complex systems. It has two levels of abstractions; one is to specify inner working of a module and other on modular interaction. G-Net follows principles of object-oriented design implementing encapsulation of a module that shields a module from external interference. It also provides a mechanism for sharing resources among modules, i.e. by G-Net abstraction. For this mechanism, G-Net is

found suitable for module-based complex system design. This paper [15] extends G-Net to make it suitable to specify webservices.

A webservice is a tupple <*NameS; Desc; URL; CS; SGN*> where

{NameS: is the name of a service used as its unique identifier.
{Desc: summarises a service functionality.
{URL: for invocation of webservice.
{CS: a set of component service a service has.
{SGN: (GSP, IS) is the G-Net modelling the dynamic behaviour of a service.

GSP represents service abstraction specifying constituting executable methods and attributes. IS represents internal structure of a service showing a set of transitions and their sequence of occurrences. Based on this idea of G-Net, the paper proposes operators like sequence, parallel, alternative, iteration and random sequence to model service composition with collaborating service components. The paper also proposes four more operators like discriminator, selection, refinement and replace. The paper has shown the possibility of service composition using these operators. The unique point the paper demonstrates is the transformation of G-Net-based algebraic specification of a webservice composition to a Petrinet model, albeit it is complex but executable as well as verifiable. Table 2 gives an overview of the existing research work and how Petrinets are enhanced to model service compositions [14].

2.3 State-Machine-Based Approach

State-machine-based software design is not new. There is a natural correspondence between system and finite state machine. A software system passed through several states, that is, represented by states in a finite state machine. System execution is considered as a dynamic behaviour of a finite state machine. A work [48] demonstrates possibility of modelling complex systems with a set of finite state of machines. Each machine represents a module. The states of a module represent a node in its corresponding state machine. A system execution is modelled by interactions among modules; an interaction is alike to a function call. It models repetitive as well as recursive calls to modules. Execution trace is a path that runs from a start state to end state of a finite state machine. It presents a hierarchical representation of state transitions to model behaviour of a complex system. Modularisation of finite state machines is a concept the paper proposes. This enables to optimise finite state machine by grouping the system states that are common to different execution paths. Further, the paper claims utility of the proposed modelling process as the modelled finite state machine (FSM) that is easily transferable to code enabling automated code creation. On considering success in finite-state-based system building, there has been research in finite-state-machine-based webservice modelling. The modelling effort here includes service modelling, verification and automatic code generation. Here, we review some works to present the utility of FSM-based webservice modelling.

Table 2 Petrinets for service modelling

Approach	Objective	References
Classical Petrinet	Propose a Petrinet-based algebra to capture the semantics of WS composition and to formally model a composite WS, which is the first step to allow the verification of the composition and the detection of inconsistencies within and among WS	[17]
Coloured Petrinet	Propose a coloured Petrinet to model types of resources managed by WSs	[18]
Time-constrained Petrinet	Propose a time-constrained Petrinet to model and analyse time-constrained WS composition	[19]
Generalised associative Petrinet model (APN) (fuzzy Petrinet)	Define automatic WS selection based on manual user specifications and using fuzzy Petrinet	[20]
Adaptation of classical Petrinet and Open workflow net	Transform a BPEL process to a Petrinet in order to allow process verification	[21–23]
Open workflow net composition net	Transform two or more BPEL processes to Petrinets and compose Petrinets in order to detect WSs incompatibility	[24, 25]
Open workflow net with coloured Petrinet	Transform two or more BPEL processes to Petrinets, compose Petrinets in order to detect WSs incompatibility, and add mediator transitions to correct partial incompatibilities among WSs	[26–28]
Classical adaptation of classical Petrinet Hierarchical coloured Petrinet	Transform BPEL or WSCI processes into Petrinets in order to verify reachability, safety and deadlock	[29–32]
Time Petrinet, adaptation of classical Petrinet	Transform WSADL specifications into Petrinets to evaluate aggregated QoS criteria of the composite WS	[33–35]
Adaptation of classical Petrinet	Generate Petrinet from OWL-S definition of WS for checking the correctness of WS specifications and the replaceability of (sub)services	[36]
Time Petrinet	Define timed Petrinet representation of WSs flow from WSDL specification	[37]
Prioritised timed extension of coloured Petrinet	Generate Petrinet from WS-CDL definition of composite WS for simulating timed or prioritised interactions among component WSs	[38]
Classical Petrinet	Propose an automatic QoS-transactional WS selection based on classical Petrinets	[39]

(continued)

Table 2 (continued)

Approach	Objective	References
Classical Petrinet	Propose an automatic QoS WS selection based on Petrinet coverability	[40]
Coloured Petrinet	Propose an automatic WS selection based on coloured Petrinets	[41]
Coloured Petrinet	Propose an automatic QoS-transactional WS selection based on coloured Petrinets	[42]
Coloured Petrinet	Propose framework for reliable execution of transactional composite WS based on coloured Petrinets	[43, 44]
Compensation paired Petrinet	Propose compensation paired Petrinet to model, evaluate QoS, verify reachability and deadlock, and control the execution	[45–47]

Webservice is a piece of code that gets executed on getting a service order. Usually, a service is hosted on the Internet and a user invokes the same through an interface. A user can be a person or even could be another service. Being invoked, different blocks of a service get executed at different contexts that make states. Context changes represent transitions. At the end of a service, result is handed over to the service invoking agency through an interface. So, a service abstraction includes both service internal module and interfacing module. This paper [8] discusses webservice design framework with the help of StateJ framework. StateJ is basically a framework to model event-based applications. It has two major components; one state transition engine and the other for event notification service. State transition engine invokes state modules at different contexts (state management service (SMS)). That can be viewed modelling of intraservice communication (request management service (RMS)), whereas event notification service (ENS) models interservice communication. StateJ uses a finite state machine to model the behaviour of a service. Authors have claimed the usability of StateJ as a natural framework for service modelling. Further, the framework provides a natural flexibility in designing a service, enabling a designer to change service component on fly. A scalable design is also possible due to StateJ. A high-level implementation of an architecture for service design is shown in Fig. 1.

Service composition is a process of generating a new service from existing services. This needs to check compatibility of constituent services. This compatibility could be at two levels. One is input and output level and the other is at conversational level. For the prior case, one service calls the other on providing expected inputs and expecting the required outputs. Later case, two services proceed with conversation while on execution. In the first case for service composition, one needs input–output match, while for the later protocol match is required for finding compatible services.

The paper [49] deals with the problem of service composition. Particularly, it deals service enactment. Service enactment means finding a service execution plan that confirms service requirements and constraints as provided by service providers and

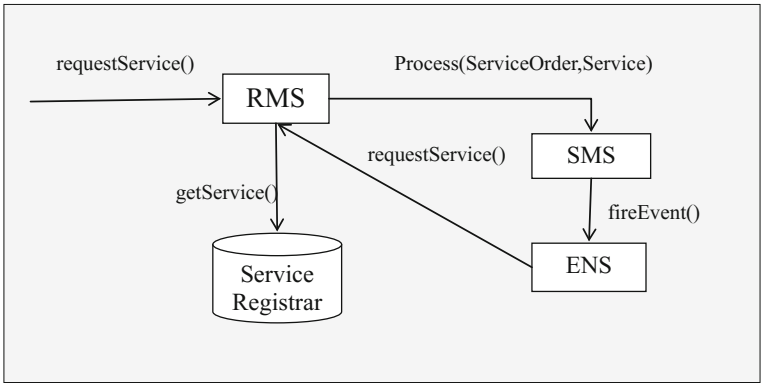

Fig. 1 High-level implementation of StateJ architecture

consumers. However, both service providers and consumers are usually not willing to expose their business details on service repository. This throws a challenge for service composition. This paper provides a mechanism that does not need stakeholders to expose their business secrets but still enables to compose a service with possible execution plan that satisfies their requirements. The proposed approach uses finite state machines to model constituent webservice operations and their interdependencies, security constraints and access control policies. It incrementally generates service enactment plan. The paper also suggests commutative encryption-based techniques to preserve privacy and security.

The paper [50] provides a formal approach for B2B collaboration among webservices. Existing webservices collaborate to provide different services that as such do not readily exist on web. In this context, study on matchmaking process takes a lead. This paper uses deterministic automata for the purpose. Before matchmaking of webservices input–output matching technique is a well-known technique. It says two services are composable when input of one matches to output of other. This standard paradigm at times may not work when there is a mismatch in their process descriptions. In spite of input–output match between two services, there could be mismatch when a service at a state needing some input from other may not get it if there is no corresponding output from the other service. This paper proposes finite-state-machine-based approach for service matchmaking. A service specification is seen as a sequence of state changes where each state is annotated with a message that has sender, message content and receiver associated with. Composition of services is viewed as intersection of state sequences services undergo during their life cycles. This is based on simple reasoning on message transmission verifying whether the right message is sent by right sender to the right receiver. In case of null intersection of state sequences of two services, service composition becomes impossible. In case of partial match, negotiation among services takes place to make the services agree to make a composed service. This idea of negotiation for composition is found new with the work.

Role of communication in service composition is further researched in [51] showing the use of Mealy machine that accepts a language meant for webservice composition. In addition to idea proposed in [50], this paper introduces a notion of global controller that keeps a watch on message passing at different webservices. In order to realise asynchrony in message passings, it introduces a notion of *prepone* operator that brings in random delay to message passing. A projection of global communication in view of a participating service presents local picture on message communication. Message passing between two webservices is formalised by *project-join* of global communication. A composite service is considered as Mealy implementation that generates conversations whose projections are consistent with participating individual webservices. However, there are some communications among Mealy machines which cannot be formalised as project-join of a regular language. Hence, the researchers propose a formalism for webservice communication that is useful for service composition. With this protocol in addition to modelling interplay between two Mealy machines, global behaviour can also be modelled. The research shows that for a given regular language it is possible to get a conversation by applying prepone and project-join closure corresponding to the set of conversations a set on Mealy machines can perform for a given language. The second result spells the conditions at which a set of conversation from a given language applying prepone and project-join closure is achievable.

Modelling and verification go hand in hand. On reviewing some of the prominent modelling techniques next we will review some important works on webservice verification and while doing so, we may touch upon some modelling issues. This may lead to some repetitions as well as additions in the next section. Readers are foretold of the fact before we proceed for review on webservice verification.

3 Webservice Verification

Software verification is a crucial activity for achieving fault-free product. There has been active research on webservice verification. State-machine-based verification is one of the techniques pursued in [52]. Two issues, i.e. communication and service module verifications are considered here. Communication between two composing services is expressed in XPath expressions. This defines guard conditions on communications. Service module defined in BPEL code is tested. For testing the later, a service BPEL code is translated to state machines that models threads execution contained in BPEL code. Communications among composing services are also modelled as state transitions. Communication conditions label state transitions. Promela is a target language to translate service implementation to finite state machine. This executable language is run on SPIN to simulate behaviour of a state machine and to identify correctness of the model. The model assumes unbounded message queue while modelling message communication. In practice, webservices are engaged in bounded communication. SPIN verifies synchronisation. SPIN can only operate with bounded message communication and partial verification of synchronisation can

also be done. The paper proves for asynchronous communication if there exists a bounded number of messages that need to be passed among webservices for synchronisation and then it is possible to verify synchronisation for the same set of messages in case of synchronous messages communication. Thus, communication protocol for a service composition is verified with the help of finite-state-machine-based models.

A formal verification approach is proposed in [53]. The approach is useful to verify workings of a service that is composed of several services. The composition follows matching of conversations among participating services. Most services perform stateful conversation so the method takes this feature while verifying working of a composed service. The novelty in this approach is its global view on modelling communication among participating webservices. They have proposed coloured Petrinet model for the purpose. This model is based on the control flow patterns of BPEL4WS where each node represents a node and an arc represents message passing. In a way, CP-net models control flow among services that make a composed service. Webservice activities are seen as message passing and action invoked thereof. On modelling, a service verification is taken into consideration. The thrust is given to protocol conformance, i.e. while putting a service in a composition the issue of conversation among services is to be relooked to ensure whether conversations among participants are well defined and so also there of actions. In addition, general criteria such as liveness, boundedness and reachability are verified by applying CP-net analysis technique. A similar work is reported in [54] that models webservice developed in BPEL to coloured Petrinet net (CPN) by mapping BPEL constituents to Petrinet nodes, transitions and colour tokens. By virtue of Petrinet concurrency, communication and synchronisation properties are being modelled. Then, using CPN tool, service behaviour is analysed and verified.

In [55] a generic framework called VERBUS is proposed for verification webservices. VERBUS framework integrates several formal verification tools like Spin and SMV. A webservice can be developed in many languages. Usually, from implementation, a formal model is to be extracted and then formal model is to be verified be it a CPN or a FSM. Thus, verification process is used to be tied with implementation language. This work is distinctly different in suggesting an intermediate specification language to which any type of implementations can be translated into. A service specification written VERBUS intermediate language is compiled to produce executable for a model checker. The paper reports verification of invariants, goals, pre- and post-conditions, activity reachability analysis and temporal properties. A logic-based approach for service verification is reported in [54]. The researchers take up service composition in two steps; one identifying services required for the purpose of a service consumer and other on selecting the right one for the identified services, while the first step ensures the functionalities and the second one ensures the quality of service. Thus, the method not only ensures selections but right selections. The first one tells signature matching and the other one as specification matching. Signature matching is based on matching of input–output parameters among constituting services, while specification matching ensures satisfaction of constraints defined over input–output parameters. For specifying, it uses both temporal logic action (TLA) and first-order logic (FOL), while the former specifies service signature the later

one specifies system behaviour. For a given query, equivalent logic expression in TLA is formulated and then services satisfying the expression are identified. Further, constraints defined on services are evaluated to choose the right ones. The key for verification rests on formulating right kind of constraints that participants of constituting services would satisfy.

Unified modelling language being de facto industry standard in software development, the work in [56] takes up the issue to show modelling as well as verifying a webservice. The process goes through several stages. Specification written in web services choreography description language (WS-CDL) is first converted to component diagram and this diagram is put through static analysis to verify some structural properties like dependency and connectedness. Secondly, a sequence diagram is generated that depicts service behaviour. In third stage, sequence diagram is used to derive state diagram that is enriched with control and data flows. Then, state diagram is translated to SMV language so that SMV model checker is used to verify different design properties that are allowed with respect to application domain. The work demonstrates the verifiability of a service specified in WS-CDL transforming into multiple UML models.

The idea verifying early in the process of development is always preferable for its utility in getting a verified correct design so that implementation will be correct provided no error has crept in during translation of a design to an implementation. The paper in [57] takes up this approach. A UML specification of a webservice in message sequence charts (MSC) is converted to finite state process (FSP) notation. And then, the specification in FSP is traced for analysis. Such notation specifies design views and then these are matched with user views. In case of match, the design is considered correct. The properties they verify include process and partners, message passing, sequence, loops and concurrency. In general, the paper proposes a technique that supports modelling, verification as well as implementation.

Petrinet-based static analysis of a webservice is presented in [58]. A process language like BPEL has been adopted for webservice implementation. BPEL provides language primitives for message passing. These primitives combined with several control features including sequential, branching, parallel and synchronisation are implemented. A BPEL process can be translated to a Petrinet. And then, making use of Petrinet analysis technique, control flow of a business process can be verified. A work towards static analysis of webservices is reported in [58].

The paper [59] presents verification of webservice choreography by analysing their cooperations. It ensures correctness in their protocols. This ensures a right business partner talking to right one on a purpose. This is ensured by their protocol matching. It also ensures to check deadlock-free communication. The approach proposed includes first characterisation of each activity with a set of operational rules. These rules clearly define permissible functional behaviour of a service. Service activities include both basic communication activities like request, reply, etc. and structured activities like flow, while, switch, etc. that model business workflow. Secondly, compatibility rules are defined to adjudge the compatibility of two processes. That means, two processes can proceed their activities in cooperation by message

passing. Then, algorithms for that check are formulated. The algorithms check both progresses of a service execution as well as their communication.

Zhao et al. [60] proposes a technique for webservice verification. It also proposes a method for generating test cases from BPEL logic so that dynamic behaviour of a service can be verified. From these, some test cases are selected with respect to counterexamples of model checking output. A service in BPEL is translated to LOTOS and labelled transition system (LTS) is translated to test and testing control notation (TTCN) behaviour tree. Keeping runtime behaviour checking in mind, the process of testing puts importance in determining frequency of testing and identification of operations to be tested. More testing should be done without interrupting service execution. A tool is developed to model BPEL with LOTUS. The EVALUATOR 3.0 tool is applied to generate LTS as output. Making use of such a tree safety, liveness and fairness properties of a service are verified.

A recent work on service behavioural compatibility and similarity study using Petrinet is reported in [61]. Seamless running of a composed service depends on compatibility of constituent services. This has been verified by matching of individuals protocols. But, this paper proves that study on termination is good enough to verify correctness of a composed service. For the purpose, services are required to be well structured. It talks of resilience by replacing a service equivalent to a failing service. Researchers define conditions to study on context-independent similarity among webservices. For the purpose, a notion of behavioural compatibility is formulated. A formalism called service workflow net (SWN) based on coloured Petrinet (CPN) is developed to model webservices. Through this formalism some important runtime behaviours like message passing, buffering and choice making are anlaysed. With the help of these studies, reachability termination and proper termination are verified. Further, for finding similarity weak soundness is considered so that conditions can be relaxed to find suitable compatible alternative matching services to replace faulty ones. A tool supporting the proposed theory for verification and similar service detection is developed.

Keeping developments in cloud computing and deployment of services on cloud there have been a concern on pace of development of service technology. In that context, the work in [62] addresses the issue of dependable services. For modelling that service behaviours are characterised to operational behaviours and control behaviours. Runtime coordination among those behaviours is achieved by message passing. Based on this modelling, an approach for service verification is proposed. The approach is based on symbolic model checking. Verification properties are extracted from control behaviours in form of temporal logic formulas. And these formulas are verified with operational behaviours. This checking is done using NuSMV model checker. Based on this approach, a set of tools are developed for service engineering confirming the possibility of automating a set of tasks including assistance in service specification, detecting design problems, debugging and monitoring service behaviours.

Resilience in webservices has become prime interest for making it useful for business houses. This has led to understanding mistakes and provisioning to overcome such mistakes. The paper [63] takes up this issue viewing webservices as event-

based transactional systems. The transactional behaviour is studied in both design and runtime to find out design errors as well as runtime errors. The method proposed includes first to translate service transactional patterns to formal expressions using event calculus. These expressions are evaluated to check transactional consistency before design and after run. Prior conditions to a transaction expressed in event calculus are evaluated. And then posterior conditions are tested using service logs. A webservice has a life cycle consisting of states, viz. initial, active, cancelled, failed, compensated and completed. It has also a set of transitions, viz. activate(), cancel(), fail(), compensate(), complete(). Pre- and post-conditions are associated with a state change. Verifiability of a service is made possible by satisfiability of these pre- and post-conditions.

Data-centric webservices require interfacing of data repository and services as per the user requirements. Further, in case of composed services, constituent services in a sequence use data. In the process, data creation, deletion and modification take place. Correctness of data usages is required in assuring correctness in data-centric service operations. A service-contract-based strategy is proposed in [64]. For a given set of data operations, a formalism is enforced. This contains conditions associated to a data operation. This is called service contract. For a given set of data, an interfacing formalism is defined so that when a service interfaces with a repository for a data operation the contract is validated. Similarly, in case of a service, service interaction contracts for data provider and consumer are defined. These contracts are to be satisfied during execution of a composite webservice. The paper presents an implementation of the proposed approach, Java modelling language (JML). Dafny and Resolve are used to specify contract specification. LML is an extension to Java programming language. Dafny has features of procedural and functional programming languages, whereas Resolver with sound mathematical foundation is used to evaluate contracts and explain later for understanding.

Runtime verification of a webservice is always important to ensure resilience in service rendering. Because, in spite of well-verified design and implementation runtime glitches may creep into put service execution in jeopardy. The issue has been addressed in [61]. Service behaviour is driven by messages. Hence, formalising message passings is the main idea proposed in this paper. Service-oriented description language (SODL) is proposed to specify protocol for message passings. Estelle-based formal model named FSM4WSR is proposed. This model is designed to capture runtime behaviour of a service. This behaviour is validated by the constraints specified in SODL.

Time is a useful and an important feature in case of service provisioning. In practice, a service is required to be done in a specific time. That could be of business interest or domain characteristics. A service execution is seen as a sequence of state changes. And each state has time constraints to be satisfied. For specifying such temporal behaviour [65] proposed a method on timed WS automata. The authors propose methods to translate description of webservice applications written in BPEL4WS to WS Timed Automata. For formal verification, tool Uppaal is used which evaluates time-related properties of a system at different states. The paper explains the proposed method by a case study on an airline travel reservation system.

Dranidis et al. [66] proposes a scheme for runtime verification of webservices. It checks whether a service is running according to specification or not. Primarily, two aspects they verify, that is, control flow and values to variables. During execution, variables must assume expected values. So, control flow must also progress as per specification, i.e. on an emanating context, control must progress resulting in successful termination of service execution. Specification on expected service behaviour is formalised by X-machines. A conversational behaviour is formalised by X-machine which is a kind of extended finite state machine. The machine also helps in monitoring verification process through model animation. The contribution of this paper includes both a method for verification and an architecture for the purpose.

Another work on runtime verification is reported in [67]. The importance of it is verification of temporal properties of a service choreography. Real-time requirements of service interactions are specified in Fiacre verification language. This language is useful to model both behavioural properties as well as timed interactions. Though in a sense the mechanism of verification is the same like other such works, still the speciality is in specifying complex requirements with the help of different formalisms. Logic-based formalism is used to express relation between two occurrences of events. The logic is built on metric interval temporal logic (MITL), a real-time extension of linear temporal logic. This enables verification of very general execution scenario, stated with both local as well as global constraints based on contexts evolving during execution.

A work that is the first in kind reported in [68] takes up both functional as well as nonfunctional requirements verification. In order to verify functional requirements a labelled transition system (LTS) from BPEL semantics is used. The system is drawn from BPEL semantics. For nonfunctional requirement verification, a roundabout strategy is used. Nonfunctional properties are translated to functional verification framework. Three nonfunctional properties, viz. availability, time and cost are considered in the work. The verification of time and cost is performed on the fly. At each state of LTS, availability and cost are calculated and tagged for verification, whereas for response time, at the LTS generation stage required response time is tagged to each state for online verification. Thus, both functional and nonfunctional requirements are verified on the fly. Thus, this is the first work at that time to take up verification of both kinds of requirements. The speciality is that the verification does not require to build a runtime intermediate abstraction to match with LTS extracted from BPEL semantics. Rather, at runtime, the extracted LTS is used for verification.

Yin et al. [69] propose a verification technique that is based on Martin-Löf type theory (MTT). Unlike other formalisation, MTT is not based on predicate logic. Instead, MTT represents predicates as correspondence between propositions and sets. A proposition is identified with a set of its proofs. A set of rules determines a proposition. Existence of a proof ensures proposition. The idea is extended for service matching. Service matching is a proposition that comes true for two compatible services. This compatibility check is governed by type checking rules. In case rules for two services satisfy then it positively verifies composition of two services. The paper discusses on verification of a service and a set of services. The verification is governed by rules for subtype, duality and consistency check.

Table 3 A summary of service verification approaches

Approach	Objective	References
FSM	Bounded message queue in communication service module verification	[52]
CPNET	Protocol conformance—liveliness, boundedness and reachability analysis	[53]
CPNET	Concurrency, communication and synchronisation verification	[54]
VERBUS	Verification of invariants, goals, pre- and post-conditions, activity reachability analysis and temporal properties	[55]
Logic based	Formulating right constraints	[54]
UML (component sequence and state diagrams)	Dependency and correctness tests	[56]
UML (MSC & FSP)	Process and partners roles analysis, message passing, sequence, loops and concurrency verifications	[57]
Petrinet	Control flow verification Static analysis	[58]
Times input and output transition system (TIOTS)	Deadlock detection and protocol matching	[59]
LOTOS and TTCN (behaviour tree)	Safety, liveness and fairness	[60]
CPNET	Reachability, termination criteria, message passing, buffering, choice making tests	[61]
Symbolic model checking	Control and operational behaviour testing	[62]
Event calculus	Pre- and post-conditions checking	[63]
Contract based (JML)	Data consistency check	[64]
X-machines	Control flow, values to variables checks	[66]
Logic based (MITL)	Temporal properties, general execution scenario analysis	[67]
LTS	Functional and nonfunctional requirements verification	[68]

The research works discussed in this section are summarised in Table 3 given below. It presents the verification objectives and associated approaches used for the purpose.

4 Conclusion

This chapter presents some early works on modelling and verification of webservices. The works are representatives of different approaches used for both modelling and verification. In this chapter, FSM, Petrinet, logic and UML are chosen for discussion based on their popularity among users and rigour for the purpose of modelling. Each one is found suitable for modelling of certain aspects of webservices.

FSM, Petrinet, logic, UML, behaviour trees, etc. are used to verify several dynamic properties of service provisioning. The service properties under study include control flow analysis and checking of system liveliness, boundedness, reachability, concurrency, constraints, deadlock, safety, fairness, data consistency, termination and temporal properties. The detailed review gives readers an insight into considering a formal model as a choice for webservice modelling and verification. Readers can appreciate the beauty of each model in specifying certain aspects of an application. So that, for a complex service requirement a designer can choose a set of relevant models useful to model various aspects of an application. At the initial stage of service engineering, such a model composition approach helps to explore uncharted dimensions of an application so, requirement engineering leads towards completeness.

All these modelling approaches being rigorous help in developing concrete algorithms for checking correctness and completeness of service models. It also helps in designing a provable service system as its behaviour is formally defined and realised in implementation.

Further, because of formal approaches for both modelling and verification of webservices, automation of both the processes has been possible. There have been several tools for formal modelling and verification. These tools with respect to different modelling approaches are also discussed.

References

1. Irum Rauf. Muhammad Zohaib Z Iqbal, Zafar. I. Malik UML based Modeling of Web Service Composition- A Survey; Sixth International Conference on Software Engineering Research, Management and Applications, 2008
2. Christophe Dumez, Ahmed Nait-Sidi-Moh, Jaafar Gaber, Maxime Wack, Modelling and specification of Web services composition using UML-S, 4th international conference on Next Generation Web Services Practices (NWeSP08), Oct 2008, Seoul, South Korea. IEEE Computer Society, 0, pp. 15–20

3. Dhikra Kchaou, WS-UML: A UML Profile for Web Service Applications, ISIICT'09 Proceedings of the Third international conference on Innovation and Information and Communication Technology, British Computer Society Swindon, UK, 2009
4. Pengcheng Zhang, Henry Muccini; Model and Verification of WS-CDL based on UML Diagrams; International Journal of Software Engineering and Knowledge Engineering Vol. 20, No. 8 (2010) 1119–1149
5. M. Emilia Cambronero J. Jose Pardo Gregorio Diaz Valentin Valero; Using RT-UML for Modelling Web Services; SAC07 March 11–15, 2007, Seoul, Korea
6. Viet-Cuong Nguyen, Xhevi Qafmolla, Karel Richta; Domain Specific Language Approach on Model-driven Development of Web Services; Acta Polytechnica Hungarica Vol. 11, No. 8, 2014, 121–138
7. W. Provost, "UML for Web services", XML.com, August 5, 2003
8. Jun-Jang Jeng, Wang-Chuan Tsai; Designing An FSM Architectural Framework for Service-Based Applications; COMPSAC, 2000, pp. 234–239
9. R. Gronmo, I. Solheim, "Towards Modeling Web Service Composition in UML", INSTICC Press, 2nd International Workshop on web services: Modeling, Architecture and Infrastructure, Porto, Portugal, April 2004
10. F. Belouadha and O. Roudiés, « Vers un modèle de spécification pour la composition de services web », Proceedings of SIIE'08, Tunisie, Février 2008
11. G. Ortiz and J. Hernandez, "Toward UML Profiles for web services and their Extra-functional Properties", IEEE International Conference on Web services (ICWS'06), 2006
12. F. Belouadha and O. Roudiés, "Un profil UML de spécification de services web composites sémantiques", CARI 2008-MAROC, pp. 537–544
13. Jia Zhang, Carl K. Chang, Jen-Yao Chung, Seong W. Kim; WS-Net: A Petri-net Based Specification Model for Web Services; Proceedings of the IEEE International Conference on Web Services (ICWS04)
14. Yudith Cardinale, Joyce El Haddad, Maude Manouvrier, Marta Rukoz; Web Service Composition Based on Petri Nets: Review and Contribution; LNCS 8194, pp. 83–122, 2013
15. Sofiane Chemaa, Faycal Bachtarzi, Allaoua Chaoui; A high-level Petri net based approach for modeling and composition of web services; Procedia Computer Science 9 (2012) 469–478
16. Y. Deng, S. K. Chang, J. C. A. De Figueiredo, A. Psrkusich, Integrating software engineering methods and petri nets for the specification and prototyping of complex information systems, in: Proc. The 14th International Conference on Application and Theory of Petri Nets, Chicago, 1993, pp. 206–223
17. Hamadi, R., Benatallah, B.: A Petri net-based Model for Web Service Composition. In: Proc. of the 14th Australasian Database Conf., ADC 2003, vol. 17, pp. 191–200
18. Zhang, Z., Hong, F., Xiao, H.: A colored petri net-based model for web service composition. Journal of Shanghai University (English Edition) 12, 323–329 (2008)
19. Yu, H., Fan, G., Chen, L., Liu, D.: Analyzing time constrained service composition based on Petri net. In: 3rd Int. Symposium on Electronic Commerce and Security Workshops, pp. 68–71 (2010)
20. Fang, X., Jiang, C., Fan, X.: Independent global constraints for web service composition based on GA and APN. In: Proc. of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC 2009, pp. 119–126 (2009)
21. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri nets. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 220–235. Springer, Heidelberg (2005)
22. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WSBPEL processes using flexible model generation. Data Knowl. Eng. 64(1), 38–54 (2008)
23. Ouyang, C., Verbeek, E., van der Aalst, W.M.P., Breutel, S., Dumas, M., ter Hofstede, A.H.M.: Formal semantics and analysis of control flow in WS-BPEL. Sci. Comput. Program. 67(2–3), 162–198 (2007)
24. Martens, A.: Analyzing Web Service Based Business Processes. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 19–33. Springer, Heidelberg (2005)

25. Xiong, P., Fan, Y., Zhou, M.: A Petri Net Approach to Analysis and Composition of Web Services. IEEE Transact. on Systems, Man, and Cybernetics, Part A 40(2), 376–387 (2010)
26. Du, Y., Li, X., Xiong, P.: A Petri Net Approach to Mediation-aided Composition of Web Services. IEEE Transactions on Automation Science and Engineering (2012) (to appear)
27. Li, X., Fan, Y., Sheng, Q.Z., Maamar, Z., Zhu, H.: A Petri Net Approach to Analyzing Behavioral Compatibility and Similarity of Web Services. IEEE Trans. on Systems, Man, and Cybernetics, Part A, 510–521 (2011)
28. Tan, W., Fan, Y., Zhou, M.: A Petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language. IEEE T. Automation Science and Engineering 6(1), 94–106 (2009)
29. Chi, Y.-L., Lee, H.-M.: A formal modeling platform for composing web services. Expert Syst. Appl. 34(2), 1500–1507 (2008)
30. Ding, Z., Wang, J., Jiang, C.: An Approach for Synthesis Petri Nets for Modeling and Verifying Composite Web Service. J. Inf. Sci. Eng. 24(5), 1309–1328 (2008)
31. Yang, Y., Tan, Q., Xiao, Y.: Verifying web services composition based on hierarchical colored petri nets. In: Proc. of the 1st Int. Workshop on Interoperability of Heterogeneous Information Systems, IHIS 2005, pp. 47–54 (2005)
32. Yang, Y., Tan, Q., Xiao, Y., Liu, F., Yu, J.: Transform BPEL workflow into hierarchical CP-nets to make tool support for verification. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 275–284. Springer, Heidelberg (2006)
33. Dong, Y., Xia, Y., Sun, T., Zhu, Q.: Modeling and performance evaluation of service choreography based on stochastic petri net. JCP 5(4), 516–523 (2010)
34. Mao, C.: Control Flow Complexity Metrics for Petri Net-based Web Service Composition. Journal of Software 5(11), 1292–1299 (2010)
35. Xia, Y., Liu, Y., Liu, J., Zhu, Q.: Modeling and performance evaluation of bpel processes: A stochastic-petri-net-based approach. IEEE Trans. on Systems, Man, and Cybernetics, Part A 42(2), 503–510 (2012)
36. Bonchi, F., Brogi, A., Corfini, S., Gadducci, F.: Compositional Specification of Web Services Via Behavioural Equivalence of Nets: A Case Study. In: van Hee, K.M., Valk, R. (eds.) PETRI NETS 2008. LNCS, vol. 5062, pp. 52–71. Springer, Heidelberg (2008)
37. Thomas, J.P., Thomas, M., Ghinea, G.: Modeling of web services flow. In: IEEE Int. Conf. on E-Commerce (CEC), Stillwater, OK, USA, pp. 391–398 (2003)
38. Valero, V., Macià, H., Pardo, J.J., Cambronero, M.E., Díaz, G.: Transforming Web Services Choreographies with priorities and time constraints into prioritized-time colored Petri nets. Sci. Comput. Program. 77(3), 290–313 (2012)
39. Blanco, E., Cardinale, Y., Vidal, M.-E.: Aggregating Functional and Non-Functional Properties to Identify Service Compositions, pp. 1–36. IGI BOOK (2011)
40. Li, B., Xu, Y., Wu, J., Zhu, J.: A petri-net and qos based model for automatic web service composition. Journal of Software 7(1), 149–155 (2012)
41. Qian, Z., Lu, S., Xie, L.: Colored Petri Net Based Automatic Service Composition. In: Proc. of the 2nd IEEE Asia-Pacific Service Computing Conf., pp. 431–438 (2007)
42. Cardinale, Y., El Haddad, J., Manouvrier, M., Rukoz, M.: CPN-TWS: a coloured petri-net approach for transactional-QoS driven Web Service composition. IJWGS 7(1), 91–115 (2011)
43. Cardinale, Y., Rukoz, M.: Fault Tolerant Execution of Transactional Composite Web Services: An Approach. In: Proceedings UBICOMM, Lisbon, Portugal, pp. 1–6 (2011)
44. Cardinale, Y., Rukoz, M.: A framework for reliable execution of transactional composite web services. In: MEDES, pp. 129–136 (2011)
45. Mei, X., Jiang, A., Li, S., Huang, C., Zheng, X., Fan, Y.: A Compensation Paired Net-based Refinement Method for Web Services Composition. Advances in Information Sciences and Service Sciences 3(4), 169–181 (2011)
46. Mei, X., Jiang, A., Zheng, F., Li, S.: Reliable Transactional Web Service Composition Using Refinement Method. In: Proc. of the 2009 WASE Int. Conf. on Information Engineering, ICIE 2009, vol. 01, pp. 422–426 (2009)

47. Wang, Y., Fan, Y., Jiang, A.: A paired-net based compensation mechanism for verifying Web composition transactions. In: 4th International Conference on New Trends in Information Science and Service Science (NISS), pp. 1–6 (2010)
48. Sukhamay Kundu; Modeling Complex Systems by A Set of Interacting Finite-State Models; APSEC, 2003, pp. 380–389
49. Basit Shafiq, Soon Chun, Jaideep Vaidya, Nazia Badar, Nabil Adam, Secure Composition of Cascaded Web Services, 8th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing, Collaboratecom 2012 Pittsburgh, PA, United States, October 14–17, 2012, 137–147
50. Andreas Wombacher, Peter Fankhauser, Bendick Mahleko; Matchmaking for Business Processes based on Choreographies; Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service
51. Conversation Specification: A New Approach to Design and Analysis of E-Service Composition; WWW2003, May 2024, 2003, 403–410
52. Xiang Fu, Tevk Bultan, Jianwen Su; Analysis of Interacting BPEL Web Services; WWW2004, May 1722, 2004, New York, USA
53. Xiaochuan Yi and Krys J. Kochut, A CP-nets-based Design and Verification Framework for Web Services Composition, Proceedings of the IEEE International Conference on Web Services (ICWS04)
54. Yabei Wang, Shangliang Pan; CPN-Based Verification of Web Service Composition Model, Proc. International Conference on Educational and Information Technology, 2010, V1-153-158
55. Jesus Arias Fisteus, Luis Sanchez Fernandez, Carlos Delgado Kloos, Applying model checking to BPEL4WS business collaborations, Proc. of ACM Symposium on Applied Computing, 2005, 826–830
56. Zhang, Bixin Li, Henry Muccini, Yu Zhou, Mingjie Sun, Data-enriched Modeling and Verification of WS-CDL Based on UML Models; Zhang, Bixin Li, Henry Muccini, Yu Zhou, Mingjie Sun; Proc. on IEEE International Conference on Web Services, 2008, 752–754
57. Howard Foster, Sebastian Uchitel, Jeff Magee, Jeff Kramer, Model-based Verification of Web Service Compositions
58. Chun Ouyang, Eric Verbeek, Wil M.P. van der Aalst, Stephen Breutel, Marlon Dumas, and Arthur H.M. ter Hofstede, Formal Semantics and Analysis of Control Flow in WS-BPEL; Technical Report, Faculty of Information Technology, Queensland University of Technology, Australia
59. Melliti Tarek, Celine Boutrous-Saab, Sylvain Rampacek; Verifying correctness of Web services choreography, Technical Report, IBISC, University of Evry, France … Find reference details
60. Huiqun Zhao, Jing Sun, Xiaodong Liu; A Model Checking Based Approach to Automatic Test Suite Generation for Testing Web Services and BPEL, Proc. of IEEE Asia-Pacific Services Computing Conference, 2012, 61–69
61. Zhuqing Li, Dianfu Ma, Yongwang Zhao, Jing Li, Qing Yang; FSM4WSR: A Formal Model for Verifiable Web Service Runtime; Proc. of IEEE Asia-Pacific Services Computing Conference, 2011, pp. 86–93
62. Quan Z. Sheng, Zakaria Maamar, Lina Yao, Claudia Szabo, Scott Bourne, Behavior modeling and automated verification of Web services, Inform. Sci. (2012)
63. Walid Gaaloul, Sami Bhiri, and Mohsen Rouached, Event-Based Design and Run-time Verification of Composite Service Transactional Behavior, IEEE Transactions on Services Computing, vol. 3, no. 1, January-March 2010, 32–45
64. Iman Saleh, Gergory Kulczycki, M. Brian Blake, Yi Wei; Formal Methods for Data-Centric Web Services: From Model to Implementation; Proc. of IEEE 20th International Conference on Web Services, 2013, pp. 332–339
65. Jia Mei, Huaikou Miao, Qingguo Xu, Pan Liu; Modelling and Verifying Web Service Applications with Time Constraints; Proc. 9th IEEE/ACIS International Conference on Computer and Information Science, 2010, 791–795
66. Dimitris Dranidis, Ervin Ramollari, Dimitrios Kourtesis; Run-time Verification of Behavioural Conformance for Conversational Web Services; IEEE European Conference on Web Services Proc. of Seventh IEEE European Conference on Webservices, 2009, 139–147

67. Nawal Guermouche, Silvano Dal Zilio; Towards Timed Requirement Verification for Service Choreographies; 8th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing, Collaboratecom 2012 Pittsburgh, PA, United States, October 14–17, 2012, 117–126
68. Manman Chen, Tian Huat Tan, Jun Sun, Yang Liu, Jun Pang, and Xiaohong Li; Verification of Functional and Non-functional Requirements of Web Service Composition; LNCS 8144, 2013, 313–328
69. YuYu Yin, JianWei Yin, Ying Li, ShuiGuang Deng; Verifying Consistency of Web Services Using Type Theory; Proc. of IEEE Asia-Pacific Services Computing Conference, 2008, 1560–1567