Formal Verification of Web Service Compositions for Developers: a Review

Rogier Monshouwer
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
r.monshouwer@student.utwente.nl

ABSTRACT

Simple web services, like weather forecasts, can be combined with other services to form web service compositions. These compositions can get complex and become impossible to be manually checked for correctness. Formal models of compositions help verify the correctness of the developed system in an automated way. Developers can use these models to discover errors in their web service compositions. However, the vast amount of research done in verification of web services makes it difficult to choose an appropriate formal model. This paper aims to clarify research done in verification techniques, especially for web service developers. The benefits for developers are restricted to verification techniques, specification translation and tooling support. This paper used a literature review on Petri nets, Process Algebra and Automata to tackle this problem, each technique represented by one paper, chosen by certain selection criteria. Using existing reviews on verification of web services, comparison criteria have been synthesised in this paper. The three papers selected are scored and compared based on these criteria. The goal of this study was to find out for each formalism what benefits towards verification it has. As a result, scores for each formalism have been given and put into a table. Interestingly, no formalism got all the points due to the focus of the paper. More papers per formalism can lead to a better comparison in this regard.

Keywords
Web service composition, verification, formal models, tools

1. INTRODUCTION

A web service is a system to support machine-to-machine communication using Internet protocols. They are the backbone of today’s Internet. Web services are split in two categories: atomic services and composite services. In composite web services small individual web services are combined to form a new service. This is called a web service composition (WSC).

Through well-defined and internationally accepted standards, the usage of web service compositions increased. Executable web service languages provide a way to describe business processes, link services together (graphically) and execute the service without any further programming. Notable examples of executable web service languages are WS-BPEL and BPMN. The former being an older standard and the latter being a standard which is becoming interestingly popular.

One WSC subject, verification, has been given a substantial amount of research. Verification in WSC is the process of guaranteeing that a composition will execute as expected, under all circumstances. Verification of web services concerns the analysis of the composition based on reachability, deadlocks, and liveliness properties. The current standards in executable web service languages cannot provide enough guarantees for correctness [10]. Standards such as WS-BPEL or BPMN provide a complete specification of the composition, but do not bring enough expressive power to guarantee its correctness. To analyse these properties, a formal model of web service compositions is needed.

Formal models (also called formalisms) can provide more scalability, reduce complexity of the composition and provide a mathematical basis that could be used to prove certain properties in WSC. Three formalism families have shown to be specifically useful to WSC: Petri Nets, Process Algebra and Automata. Each of these models with its own benefits towards the verification of WSCs.

However, the focus in current research is too much on the theory of formal models. Although several studies outlined algorithms to transform web service standards like WS-BPEL into formal models, the different models that exist have different functionalities concerning verification. Yet, web service developers can greatly benefit from verification, since the models can help discovering errors before the implementation phase, saving much valuable time. So, design and implementation developers benefit from thorough verification.

Currently there is no overview for developers seeking to verify their web service composition with formal models. Although much research has already been done and many papers are available. Consequently a literature study can be done to create such an overview of the existing formalisms, showing their beneficial properties to web service developers and helping them to choose the appropriate model for the verification of their web service composition. The three most important formalisms are evaluated. However, the amount of studies covering formal models is huge, so only one paper for each model is used in this evaluation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

27th Twente Student Conference on IT July 7th, 2017, Enschede, The Netherlands
Copyright 2017, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

1https://docs.oasis-open.org/ws-bpel/2.0/OS/wsbpel-v2.0-OS.html
2http://www.bpmn.org
The three papers are selected through incremental refinements on the search results. Then the papers are valued on criteria representing the benefits to web service developers. These criteria are synthesised from other surveys concerning verification of web services and are used as comparison criteria. For each comparison criterion, a three point scoring system has been defined. These scores help to effectively compare the three different papers, resulting in a table. The scores are compared to one another to show the differences. The goal of this study is to produce a clear and sound comparison between the formalisms.

1.1 Problem statement

Verification of web service compositions has certain benefits for developers. The detection of errors in the design phase increases productivity in the implementation phase. Consider the example shown in Figure 1.

![Simple web service diagram](image)

Figure 1. Simple web service

This is an example of a simple web service composition that represents a bank transaction between two parties. It uses several atomic web services to do certain actions related to the transaction. Notice that the “Decrease balance sender” and the “Increase balance receiver” services are executed in parallel, but both need to be finished before the transaction is completed. Verification of this web service thus should check for all situations whether the parallel services are successfully stopped before reaching a stopping point. This is just a simple correctness check on the model, but more verification techniques are possible. A few factors are important for developers that seek to verify compositions like this. These factors determined the research questions for this paper.

1. What are the existing solutions to formal modelling of web service compositions?

Answering this question narrowed down the available papers for the research. A maximum of three solutions have been chosen to allow detailed answers to the rest of the questions.

2. How do the different models compare to one another with respect to their verification methods?

Knowledge of available verification methods helps developers to choose the appropriate model. Not only is important which verification methods the formal models provide, also how scalable they are, since compositions are likely to be complex.

3. How do the different models compare to one another with respect to the mapping of executable web service languages to the formal model?

Mapping of executable web service languages to formal models have two directions. From executable language to formalism means all the constructs get a formal counterpart. This mapping of executable web services to a formal model is essential for developers because it allows the verification to be executed on the existing designs. It also enables the developers to keep their existing design solutions instead of switching to a new design language. There is also the mapping from a formalism to an executable language. This means a web service can be defined and verified in the formalism. Then the web service is translated into an executable process with proven correctness.

4. How do the different models compare to one another with respect to ease of learning and quality of tools available?

The skill required is important for developers and companies to know, because it tells them how much time it takes to adopt verification in their existing web service design phase. Better quality tools result in faster and more reliable results.

This paper is structured as follows: Section 2 analyses the similar studies done with respect to our research goal. Section 3 describes the approach of this paper in detail. It also shows the selection criteria and the comparison criteria. Section 4 presents the three selected papers used. Section 5 gives a justification of the results. Finally, Section 6 presents the conclusion of this paper.

2. RELATED WORK

Chan [4] showed the three major families of formal models for web services. The goal was to answer whether it is possible to bundle different kinds of models together, to have all the benefits of each model. Chan also found that the current tools support for those formalisms is lacking by showing what could be improved. However, this research only focussed on the formal models and did not cover any translation between executable WSC specification and formal models.

The survey of Hai Huang et al. [7] discussed different perceptions researchers have towards web services. They showed different verification techniques from various perceptions bounded by model checking. These techniques showed different tools and the paper thus works as a guide towards picking model checking techniques. Its scope was limited to automata and so the guide is limited.

The study of Morimoto [10] reviewed the three different families of formal models used in WSC verification. The aims of verification were shown, together with some information to translate BPM into those formal models. This study briefly describes what sort of verification is possible with each formal model and what tools are available. As it only presented possibilities, it does not truly compare formalisms.
The research of Danapaquiane et al. [5] showed the lack of non-determinism in existing formal models of web services. They compared five different studies, all using deterministic AUT, to their own work. They defined a non-deterministic Muller automata which could be converted into a Coloured Petri Net. They stressed the importance of the state space explosion problem seen in large and complex automata. The scope of this survey is limited to automata and the paper did not cover the translation between executable WSC specification and a formalism. Tool support in the literature was only briefly mentioned for these papers that used a tool.

The study of ter Beek et al. [14] aimed to assist WSC designers and developers by choosing the proper technology to be used. They reviewed two syntax-based and two semantics-based WSC standards. The criteria for comparison were connectivity, correctness and Quality of Service. In addition to the standards, ter Beek et al. also covered three formal models (PA, automata and PN) on these criteria. The papers related to those formalisms were only discussed briefly with respect to verification methods and tool support.

In the survey of Campos et al. [3] existing formalisms are compared on three criteria: the composition models, the classes of formalisms and the purpose of the formalism. They argued that the composition models for orchestration and choreography are the most popular. PNs and PAs are the formalisms mostly used, independent of the composition model. The purpose of the studies found during their research were mostly about mapping WSC specifications to formal models. Tool support was not covered by this survey and the verification properties were not analysed in depth.

The paper of Milanovic et al. [9] explains the basic syntax and behaviour of BPEL and other web service languages, including the use of Pi-calculus, PNs and Finite State Machines in the context of WSCs. This study briefly reviews some properties of the languages, like support to nondeterminism in existing formal models of web services composition and scalability. In this paper, scalability of the language means the amount of effort needed to describe a complex composition. Scalability issues of the language concerning computation time when verifying the correctness of the composition was not taken into account.

3. APPROACH
3.1 Background
All the models mentioned below are formal, which allows for exact analyses of certain properties.

Petri nets.
A Petri Net (PN)\textsuperscript{3} is a modelling language. It is used for the modelling of concurrent and distributed systems. It can be represented as a bipartite graph in which nodes are called places and transitions. Edges between the places and transitions depict the relation between them and often have a label attached to define a condition. An example of a PN is given in section 4.3.

Process algebra.
The family of Process Algebras (PA)\textsuperscript{4} contains similar approaches in modelling concurrent and distributed systems. They all use primitives and operators to depict processes, use message-passing communication when interacting and define algebraic laws for process operators. Examples of PAs are CSP, CCP and LOTOS. They allow exact communication and synchronisation between processes. In addition, they also provide strong process equivalence checking. An example of a PA process is given in section 4.1.

Automata.
The theory of automata\textsuperscript{5} is a modelling language concerning abstract machines. It is a popular language to abstractly model computer systems. A widely used example of an automaton is the finite state machine. This automaton has different states, which can be reached by transitions. The task of an automaton is to calculate, given an initial state and a list of transitions to a final state. Many variants to this model have been defined, to allow modelling of concurrent systems. An example of an automaton is given in section 4.2.

Model checking.
Model checking is a way to check if a model meets its specification. The models defined above can be used for model checking. Developers can use temporal logic languages like linear temporal logic (LTL) and computation tree logic (CTL) to verify existential or universal properties of the model. Absence of deadlocks is one of these checkable properties. Usually, model checking is done by creating a state space containing all possible states of a model. In concurrent systems, this state space can be huge, so model checking is known to be a time consuming task.

3.2 Steps
The research questions have been answered by doing a literature study. This was a (semi-)systematic review based on the protocol defined by Kitchenham [8]. This protocol states how systematic reviews should be done in the Computer Science field. It is a very intensive process, so the product of the review is thorough and reproducible. It involves a strict protocol of time consuming tasks such as data quality assessment and meta-analyses. However, this protocol goes beyond the scope of this research. Therefore, some generalisations and adjustments have been made in the protocol used for this study. The review consists of three phases. The first phase is the selection phase, in which an initial set of papers have been refined into three papers. Three papers allow this study to answer the criteria in sufficient detail and discuss the most used web service composition formalisms. In the second phase comparison criteria are gathered and synthesised. The second phase is the gathering and synthesis of comparison criteria. These criteria are based on related work and have helped answer the research questions. Each criterion has a three point scoring mechanism, in which it can receive a (+), a (+/-) or a (-). For each criterion, the requirements for these scores are defined. The third phase is the data extraction phase. In this phase, the comparison criteria are applied to the three papers. For each criterion, each paper received a score and justification of this score. The results represent the answers to the research questions.

3.3 Literature selection
The first step in literature selection is the identification of all research on the topic. The sources have been digit-
tal libraries from ACM\textsuperscript{6}, Elsevier\textsuperscript{7}, Springer\textsuperscript{8} and IEEE\textsuperscript{9}. Those libraries are searched using search engines like Scopus\textsuperscript{10}, Web of Science\textsuperscript{11}, Google Scholar\textsuperscript{12} and the search engines available on the website of the library.

Our literature search started with the following query: ((web OR service OR work-flow OR flow) AND verification AND composition). This query has been executed on the various databases. To restrict the resulting papers to three, several refinements based on title and abstracts have been done. First of all, the paper has to study static verification of web services. Dynamic verification is another problem and cannot be compared. As a second selection criterion, the paper has to address syntax-based web services, since current standards are syntax-based. Like dynamic verification, semantic based web services are not in the scope of this study. The third selection criterion is that the focus of the papers should be entirely on machine-to-machine communication. Business processes and workflows often involve human actors and some standards have incorporated this (e.g., BPEL4People), but this makes the automatic translation into a formal model harder. As a penultimate selection criterion, for each of the formal models, only one paper has to be selected. This highlights the differences between the models with respect to their verification. To finalise the selection process, the remaining studies have been sorted according to citation count and the paper with the highest amount have been chosen. Highly cited papers potentially have a better quality and have a high impact on the field of research. This concludes the selection process.

3.4 Comparison criteria

The comparison criteria in this study have been a tool to guide the interpretation of the findings and reflect its relevance on the research questions. Six criteria were defined and an abbreviation is used in the table presented later. For each criterion, its relation to the research goal is explained, the related work where it is based on is mentioned and scores are defined to be able to evaluate and compare each paper.

SE: The methods for handling state explosion.

This criterion helps answering the second research question. State explosions are one of the issues surrounding formalisms. Since the verification involves model checking, a state space has to be generated. Large compositions lead to an exponential number of states. This scalability issue affects the verification methods as they become time expensive. If the model has methods to handle the state space problem, it becomes more effective in usage. This criterion was identified in [10, 5, 4]. A (+) is given to the formalism if it uses techniques to avoid creating and searching the state space or when the state explosion problem does not have role in the formalism. A (+/-) is given to the paper if it is aware of the problem and proposes solutions to this problem as future work. A (-) is given if the paper does not mention this problem.

VP: The amount of different verifiable properties are referenced on the formalism. This can be categorised in model-checking (reachability, soundness/correctness, state-space analysis, invariants, deadlock absence, liveness, emptiness, boundedness, coverability, connectivity), simulation (equivalent analysis) and Quality of Service (duration, response time, throughput, price, timeout).

This criterion helps answering the second research question. The properties mentioned above are discussed in [3, 14, 10, 5]. The number of properties mentioned by the paper is the score on this criterion. New verification methods are added to the score.

CT: One-way or two-way complete translation from an executable composition language to the formal model.

This criterion helps to answer research question three. The paper can have options on what to specify on translation. Developers of web services use executable composition languages and it would be convenient if the complete translation was proposed. The translation is defined as complete when all language elements can be translated. A one-way translation can be from executable language to formalism or the opposite. Moreover, a two-way translation can speed up the verification process, as the adjusted formal model can be checked immediately and then translated back into the executable language. This criterion was also mentioned in [3, 14, 10, 7]. A (+) score is given if the formalism provides a complete two-way translation between executable language and the formal model. A (+/-) score is given if the paper provides either an incomplete two-way translation or a complete one-way translation. A (-) is given if the paper provides less or equal to an incomplete one-way translation.

DE: The amount of detail of the examples and their explanation when discussing different structures in the formal model.

The amount of detail of the examples and their explanation when discussing different structures in the formal model. (DE) This criterion helps to answer research question four. The formal models are likely to be complex and use many different structures. The usage of examples is important to visualise and understand the information that is presented. The explanation is key to understanding those examples, as reasons can clarify and make the reader think in a similar way. This criterion was also found in [14, 7, 4], but in a slightly different way. They suggested that the learning curve and intuition of a formal model is important, but the use of examples is more important when building a tool according to the author. A (+) score is assigned if the paper has extensive examples explained in detail. A (+/-) is assigned if the paper has some examples with detailed explanation or extensive examples with limited explanations. A (-) is assigned if the paper has less than or equal to limited examples with limited explanation.

TFV: The reference of a (user-friendly) tool for automatic translation and (automatic) verification of the web service composition.

This criterion helps answering the fourth research question. A tool can help the web service developer verify the composition. Automatic translation makes the verification a less time consuming task and also excludes any human
error made during translation. When the tool also provides automatic verification of some properties, the productivity of the developer increases even more. This criterion is also mentioned in [3, 14, 5, 4]. A (+) score is assigned if the paper referenced a tool with automatic translation and verification of properties and integration with other software tools if possible. A (+/-) score is assigned if the paper referenced a tool with manual translation and automatic verification. A (-) score is assigned if the paper referenced a tool with manual translation and verification or did not even mention a tool.

4. RESULTS

The initial query resulted in 1944 papers. These papers were selected based on the selection protocol described earlier in section 3.3. After the literature selection process, three papers were identified as the best papers to review in detail. For PA, the paper of Ferrara [6] was chosen. It was published in 2004 and was one of the first approaches of using PA to verify web services. It presents a framework for the design and verification of WSC using PAs. In the paper, Ferrara used LOTOS as PA to illustrate a two-way mapping between BPEL and PAs. For automata, the paper of Bentahar et al. [2] was chosen. This paper was published in 2012 and focuses on the verification of web services. Instead of providing a mapping between an executable web service language and automata, it starts with the assumption that a web service can be defined as an automaton for operational behaviour and an automaton for control behaviour. Bentahar et al. show several techniques for fast and complete verification and proved this by building a tool that implements this theory. For PNs, the paper of Ouyang et al. [12] was chosen. This paper was published in 2007. It takes a detailed angle at the translation between BPEL and low level PNs. They use their complete translation to build a tool that translates and verifies a complete BPEL composition on some interesting properties.

4.1 Process Algebra

Ferrara uses LOTOS, a PA, to formally model web service compositions. They use the definition of LOTOS standardised by the ISO13, which combines data and operations with processes.

The paper shows an abstract framework of BPEL constructs with equivalent LOTOS code. Figure 2 is a snippet of this schema.

```xml
<sequence...>
  <... act1...>
  <... act2...>
</sequence>
</flow>
</case>
</otherwise>
<... act2...>
</... act2...>
</switch>
```

Figure 2. BPEL constructs in Automata

The state explosion problem is not addressed in this paper. Although it is a problem in LOTOS [1], the paper did not address the issues when discussing the model checking verification. Probably due to the lack of a complete translation with a complete example and the lack of a tool, the researchers did not identify this issue and thus did not give a solution to the problem.

Temporal logic model checking is available by using LOTOS, so reachability, liveness, deadlock freedom and soundness/correctness checking can be verified by the developer. In addition to this, bisimulation analysis can be done, which is an equivalence analysis using behaviour as criteria for equivalence. Data type checking is possible by using LOTOS. This means that it allows the verification that each message between the services is the correct data type. Black box testing is available using LOTOS, so the user can input a lot of different messages with data and check the output. In total, (7) verification techniques are mentioned.

The paper proposes a two-way mapping between LOTOS and BPEL. All language constructs have a BPEL specification. Using LOTOS as design language is possible, but special guidelines need to be followed to obtain a simple automatic translation. This is not the case when designing with BPEL. The mappings of basic constructs, dynamic behaviour, data definition, data handlers, fault handlers, compensation handlers and event handlers are explained. An example of a translation is given, but the full translation is missing. The paper presents general guidelines for PAs to be translated.

Every aspect of BPEL has been discussed and a simple example is described. These examples are explained in detail. The examples are written in pseudo-LOTOS code for structure only. The paper does not show an example of a BPEL process translated into LOTOS. It does reference examples of WSCs modelled in LOTOS.

In the introduction, the standard verification tool for LOTOS is briefly mentioned. When the WSC is translated to LOTOS, it probably can be verified using this tool. Since the translation is not fully specified, no automatic translation is possible and thus no automatic translating tool is presented.

4.2 Automata

The paper from Bentahar et al. uses automata as a formal model for web service compositions. They propose their

---

The paper introduces a new tool for semi-automatic verification, which alleviates the need for state space traversal by using binary decision diagrams and Boolean functions. The approach of this paper also reduced the size of the Kripke model for further optimisations during model checking.

Model checking is possible with extended soundness checking. Model checking can verify deadlock absence, reachability, liveness and soundness. The control behaviour allows for very specific temporal logic formulas and thus offers extra soundness checking. Moreover, the modelling of control behaviour enables checking all possible interactions within the composition. This is known as the completeness property. So (5) verification techniques are mentioned in this paper.

The paper splits a WSC into two types of behaviour: operational behaviour and control behaviour. Operational behaviour is how web services should interact and control behaviour is how the sequence of interactions should be in the composition. It does not provide a translation from an executable language into these behaviours. The paper presents guidelines for formalising a WSC into these automata. For operational behaviour the paper uses extended finite state machines and for control behaviour alternating Büchi Automata. The translation from the operational automaton to a Kripke model is defined. Developers have to translate their executable web service program manually into these models or create their own automatic translation tool.

Throughout the paper, a ticket reservation system is used. Each step of the translation process is explained using this example. This eases the learning curve of this approach, since the verification requires a lot of steps. The ticket reservation example is not supplied with detailed explanation, which would be expected, since the transformations of the examples were not trivial.

The paper introduces a new tool for semi-automatic verification. Using the operational behaviour depicted as an automaton and the control behaviour as CTL and LTL formulas, the tool checks whether those formulas hold. It provides no integration with an executable language tool. It does integrate with the model checker. The tool expects that the user extracts CTL and LTL formulas from the control behaviour.

4.3 Petri Net

The paper of Ouyang et al. used PNs to formally model web service compositions. They used the PN definition of Murata [11] and Peterson [13] which allows labelled and unlabelled transitions.

The paper used visual representations of the translation from BPEL to PNs. Figure 4 shows part of those translations, with BPEL code on the left and an equivalent PN on the right.

Figure 3. Ticket reservation system in automata

The approach used translation invariants to alleviate the creation of the state space. Using transition invariants, the state space does not need to be traversed and therefore the scalability of this approach is greatly increased. The approach optimises verification even further by using transformation rules to reduce the size of the PN. The study addressed the problem of state explosions and used several techniques to handle the problem.

They used model checking verification techniques. This includes verification of soundness, liveness, deadlock absence and reachability. They can be used for conformance checking on process traces, which can be used to analyse execution orders of BPEL processes. Another type of verification technique introduced in this paper is based on a BPEL rule. The BPEL specification states that processes must not simultaneously enable two equivalent “receive” activities. The tool used in this paper has the ability to detect violations against this rule. Other standard PN analysis techniques, such as boundedness checking, can be used, but were not mentioned in the paper. In total, (6) techniques were mentioned in the paper.

A one-way complete mapping of BPEL to PN is provided. The appendix covers the entire BPEL specification. This includes the mapping of control links and communication in BPEL to PNs. These structures can be found in the class of synchronising workflows, which the authors studied before using PNs. Since all elements of BPEL are covered, the translation is defined as one-way complete.

The extensive use of examples and the full translation in the appendix makes this approach easy to learn. The paper first describes the BPEL language, then the mapping onto PNs, next the analysis of the PN and finally the evaluation of the method. In the mapping section, every aspect is supplied with an example and an additional BPEL process example translated to PN and explained.

This paper introduces two new tools called BPEL2PNML and WoBPEL. The first tool translates BPEL specifications to PNs using the translations discussed in the paper. The second tool analyses the BPEL-translated PN by applying the various methods proposed in the paper. As they directly use the BPEL files, the tools are user-friendly. WoBPEL includes automatic verification analyses, so the entire verification process is automatic.
5. DISCUSSION

The results are presented in Table 1. No formalism has a full score on all the points. Each paper had a different focus in the verification process, which explains why no formalism scored fully on all criteria.

<table>
<thead>
<tr>
<th>Table 1. Overview of scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Algebra</td>
</tr>
<tr>
<td>Automata</td>
</tr>
<tr>
<td>Petri Net</td>
</tr>
</tbody>
</table>

The scores will now be discussed column-wise from left to right.

The low score of PA was expected as the paper of Ferrara only provided a framework to design and verify web service compositions using PA. Automata got a (+), because the paper of Bentahar et al. provided multiple solutions to the problem. The scope of the paper was on the theoretical aspect of verification, so it was expected that they encountered the problem and tried to solve it. PN got a (+/-) because the paper of Ouyang et al. provided multiple solutions to the problem. They chose PNs, because they knew the problem existed and PNs could provide a solution. The last two papers paid more attention to the problem of state explosions. Those two papers were written around the same period, when state explosions got more attention than when the first paper was written. Automata and PNs are the best choice during verification for developers, because of their time saving capabilities.

The paper of Ferrara referenced 7 verifiable properties. PA is often used for web service composition, because it provides elaborate verification methods. The paper of Bentahar et al. referenced 5 model checking techniques. Automata are widely used in model checking systems so it was already expected that this paper had focused on the model checking techniques for verification. The paper of Ouyang et al. referenced model checking techniques and some BPEL specific techniques to a total of 6 verification techniques. The paper focused on the mapping of BPEL to PNs, so it could be expected that it would provide some extra BPEL specific verification properties. The paper on PAs performed rather well on the number of properties, given that this paper is the oldest. This is probably the reason why PAs are used often for web service modelling. The paper on PNs introduced some specific verification properties for BPEL web services, which could be convenient. PA offers the most properties for verification and thus PA is the best general purpose verification formalism for web service developers.

The paper of Ferrara provided a framework for translating BPEL web services to PAs. BPEL was only introduced a few years before its publication, so the authors possibly did not have enough time to provide a full translation. They did introduce the concept of two-way translation which can bring significant benefits to a developer when verifying compositions and so it received a (+/-). The paper of Bentahar et al. provided only guidelines to translate web service compositions into automata. The scope of this paper was not to study the translation part but to focus on the theoretical implications of verification and so automata receives a (+). The paper of Ouyang et al. provided a complete translation from BPEL to PNs. This was the main goal of the study and that is why they published the complete translation in the appendix, but PN only scored (+/-). Ouyang et al. and Ferrara scored the same, because Ouyang did not provide or mention the possibility of a two-way translation, which, as Ferrara stated, has some significant benefits. If Ouyang had mentioned this possibility, PN would have scored a (+) as stated by the comparison criteria. Still, developers using BPEL in their web service compositions will find PN the most valuable of the three, since both PA and automata lack a precise translation and developers have the responsibility to define the translation themselves.

The paper of Ferrara provided each part of the specification of LOTOS with an example. Despite that the examples were well explained, they were very limited and so PA scored a (+/-). The paper of Bentahar et al. provided an example throughout the entire paper. Explanations for their examples lacked in this paper. The examples would be more effective if the reasoning behind them was described and so automata received a (+/-). The paper of Ouyang et al. was extensive in the use of examples and supplied detailed explanations with them, making it easier to follow. The paper of Ouyang et al. excelled at this and therefore only that paper receives a (+) score. For developers, PNs would be the easiest to learn, because of the examples given in the paper of Ouyang et al. PAs and automata could require more time and other studies to learn the language.

The paper of Ferrara only referenced a tool for PA and did not discuss this in any detail. The goal of that paper was to propose a framework for web service formalisation and verification and so a tool was not necessary to reach that goal. Therefore, PA received a (-). The paper of Bentahar et al. provided a semi-automatic verification tool for the model checking of web services. The scope of this paper was not specific executable web service verification, but rather a discussion of a general purpose web service verifier. This explains why the study did not discuss an automatic translator or fully automated the verification process and so automata scored (+/-). The paper of Ouyang et al. provided an automatic translation and verification tool specifically for BPEL web services. This study was directly aimed at the translation from BPEL to PNs for analysis purposes, so it is logical that the tool they built does both translation and automatic verification. The tool of Ouyang et al. is the most practical of the tools. It provides both translation and verification, and the latter can be done mostly automatically and so PN receives a (+) on this criterion. The best tool for web service developers is the one provided with the paper of Ouyang et al., since every step is automatic. The process for automata requires more work and for PAs, more research needs to be done to find the appropriate tool for web service verification.

None of these three formalisms scored full points, while the papers did not mention any issues concerning the comparison criteria. So other issues affecting the results might have happened during our study. At two points in this study choices have been made. These choices are likely to have affected the outcome. First, in the selection process the paper with the highest citation count was chosen for this research. This was under the assumption that frequently cited papers are overall valuable to the field. However, it could be argued that the papers were often cited only because they covered some aspect in detail. Second, the selection of criteria has affected the results. These were created using related work assuming the related work had a sound overview of the literature on web service. If the related work missed some aspects on web service verification, it cascaded onto this study and thus
affected the results. These two points could have created a bias towards the results of this review.

6. CONCLUSION
The scope of this study was limited to three papers, one for each formalism. As a result, no formalism scored full points on the criteria. PA scored well on verification methods and translations, but it scored low on tool support and the state explosion problem. Automata scored well on the state explosion problem, but failed to supply a translation and scored average on the other criteria. PN scored high on the detail level of the examples, the tool support and the state explosion problem, but missed points on translation and on verification methods. Given this research, PN appears to be the best choice for verifying web service compositions. However, since the scope of this research was limited to three papers, the results can be improved. To have a better understanding of the potential of a formalism, more papers have to be chosen in the selection process. When more papers are evaluated, the scores will be more representable and a better comparison can be made. To produce an even more sound comparison for verification formalisms, other families of formalisms can be incorporated in the study.

As pointed out in the discussion, the selection process could have created bias towards detailed studies. An interesting study would be to see what the effect of different selection criteria will have on the comparison criteria present in this paper. Besides, more research is needed on the translation of execution standards such as BPMN to a formal model. More research is also needed on bundling formal approaches. As shown in the paper, the formal methods had different verification techniques and optimisations, so bundling these formalisms in a tool would produce a complete verification tool, thus alleviating the need of choosing a formal model for verification.

7. REFERENCES