Automata-Based Verification of Security Requirements of Composite Web Services

Hongyu Sun[1], Samik Basu[1], Vasant Honavar[1] and Robyn Lutz[1,2]
[1]Department of Computer Science, Iowa State University
Ames, IA, 50011-1040, USA
[2]Jet Propulsion Laboratory/Caltech
{sun, sbasu, honavar, rlutz}@cs.iastate.edu

***Abstract*— With the increasing reliance of complex real-world applications on composite web services assembled from independently developed component services, there is a growing need for effective approaches to verifying that a composite service not only offers the required functionality but also satisfies the desired non-functional requirements (NFRs). In high-assurance applications such as traffic control, medical decision support, and coordinated response to civil emergencies, of special concern are NFRs having to do with security, safety and reliability of composite services. Current approaches to verifying NFRs of composite services (as opposed to individual services) remain largely ad-hoc and informal in nature. In this paper we develop techniques for ensuring that a composite service meets the user-specified NFRs expressible in the form of hard constraints e.g., "response time has to be less than 5 minutes." We introduce an automata-based framework for verifying that a composite service satisfies the desired NFRs based on the known guarantees regarding the non-functional properties of the component services. We further show how to improve the efficiency of verifying that a composite service indeed satisfies a desired set of NFRs by: (i) Exploiting information about the applicability of specific NFRs (e.g., security) only to certain subsets of the component services that make up a composite service to minimize the verification effort and (ii) Identifying inconsistencies between NFRs with overlapping scopes. We illustrate how our approach can be used to verify the security requirements for an Emergency Management System. We also show how the approach can be used to verify whether a composite service satisfies any desired set of NFRs that can be expressed in the form of hard constraints of a quantitative nature.**

Keywords-Composite Web Service, Security, Verification, Quality of Service

I. INTRODUCTION

As web technologies become increasingly widespread, there is a proliferation of independently developed web services in many application domains. Complex real-world applications typically rely on composite web services assembled from multiple independently developed composite services. Consequently, a variety of approaches have been developed for assembling composite services that satisfy the user-supplied functional specifications [1, 2]. Functional requirements (FRs) describe how the composite service ought to process its input so as to generate the desired output. A number of techniques are available for verifying whether a composite service satisfies the user-specified FR [1, 2, 9].

However in many applications, a composite service needs to satisfy not only the desired FR but also the user-specified *non-functional* requirements (NFRs) [25]. NFRs typically specify constraints that must be met with respect to the security, safety and reliability of composite services. NFRs can be hard constraints or soft constraints. Hard constraints refer to constraints that must be satisfied e.g., "response time has to be less than 5 minutes". Soft constraints on the other hand specify user preferences over non-functional attributes e.g., "the lower the cost, the better". NFRs that specify quality requirements e.g., with respect to response time, are often also called Quality of Service (QoS) requirements [10].

Ensuring that a composite service satisfies not only the desired FRs but also NFRs is especially critical in the case of high assurance applications such as traffic control, medical decision support, and coordinated response to civil emergencies [27]. Consider, for example, an Emergency Management System (EMS) [6]. The key FR of an EMS (see Fig. 1) is to *dispatch ambulance(s), fire truck(s) and police to a location upon request using available resources*. An EMS consists of several components: the *Scheduler* Service that takes the request messages from the field officers' mobile terminals; the Emergency *Resource* Services (e.g., Police.A and Police.B) that interact with the resource databases and manage the local resources of ambulances, fire trucks and police; and the *Dispatcher* Service that interacts with ambulances, fire trucks and police cars, and dispatches them to the target location.

An EMS is a high-assurance system because failure to meet its functional (e.g., dispatch ambulance to an accident victim) or non-functional requirements (e.g., keep patient information confidential) can have disastrous consequences. The reliability of the system, the availability of the services, the effectiveness of resource allocation and the security of communications are crucial to public safety. For example, messages in an EMS must be secured against malicious eavesdropping, interception and falsification before deployment. Hence, security NFRs for an EMS might require that the messages in different scenarios be sent at different encryption levels depending on the type of emergency incident. For national security related incidents, messages and service operations may need to be protected with stronger encryption than in the case of civilian incidents. An example of a security NFR for an EMS is that *a request to dispatch police shall be processed using highly-encrypted message paths*.

Security guarantees for encryption and authentication of the messages for *each* service are typically specified using the web service security policy (WS-Security and WS-Policy) which is included in the WSDL specification for the service [20]. The security NFR for the EMS, however, applies to the *composite* service which consists of the multiple component services that make up the EMS. This presents us with the challenge of verifying that the security guarantees available for the individual component services that make up the composite service indeed satisfy the security NFR of the composite service. For a composite web service, the security requirements can apply either globally (to all services in the composite web service) or locally (to more than one service operation but not to all). Service operations are the external functionalities of a component service defined in their public interfaces. A service operation takes an input, takes actions on the input and returns the output. A component service can have one or more service operations.

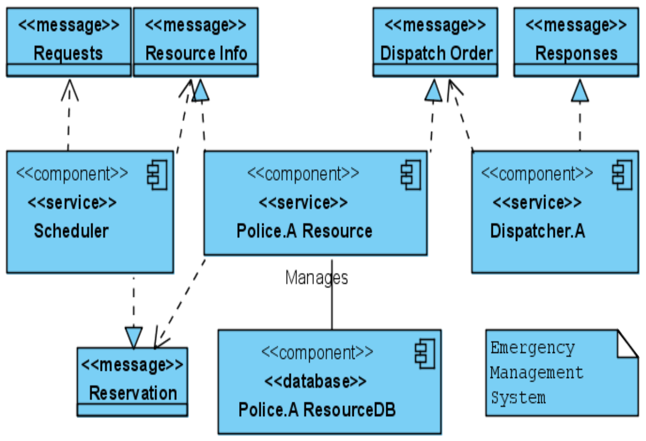


Figure 1. An illustrative example: excerpt of a web-based Emergency Management System

Other similar emergency management systems in existence include the Incident Command System of the U.S. Federal Emergency Management Agency (FEMA) and the National Incident Management System (NIMS) for major, multi-site incidents [29]. Many incidents in emergency management systems have historically been caused by incorrect request or dispatch data (e.g., the London Ambulance System in 1992 and the Australian Emergency System in 2006) [28].

Current approaches to verifying non-functional requirements (NFRs) of composite services (as opposed to individual services) remain largely ad-hoc and informal in nature. In this paper we develop techniques for ensuring that a composite service meets the user-specified non-functional requirements expressible in the form of hard constraints. We introduce an automata-based framework for (1) representing NFRs that can be expressed in the form of hard constraints and (2) verifying that the composite service satisfies the NFRs based on the known guarantees regarding the non-functional properties of the component services. We illustrate how our approach can be used to verify the security requirements for an emergency management system. However, the proposed approach can be used to verify whether a composite service satisfies any desired set of non-functional requirements that can be expressed in the form of hard constraints of a quantitative nature.

The proposed approach to static verification of NFRs of composite services incorporates two novel elements:

1. **Scoping of NFRs**: In many situations, specific NFRs (e.g., with respect to security) may be applicable only to specific subsets of the component services that make up the composite service. Moreover, the subset of services that need to satisfy a NFR may differ across the different NFRs (e.g., response time, security). Existing quality of service models for composite web service are not expressive enough to represent NFRs that apply to specific subsets of the services that make up a composite service [10, 11, 12]. To allow modeling of NFRs that apply to different subsets of components of a composite service, we introduce the notion of scope. The increased precision in the description of NFRs enables more efficient verification of NFRs.
2. **Consistency checking of NFRs**: NFRs with overlapping scopes can be inconsistent (and hence unsatisfiable) in the case of some candidate composite services. This scenario is not uncommon in settings where attempts to achieve one NFR (say, with respect to security) may rule out the achievement of another NFR (say, with respect to response time) [3]. Hence, when the scopes of different NFRs overlap, checking the consistency of NFRs can help avoid effort that would otherwise be wasted in exploring composite services that would violate the NFR constraints. Our automata-based model permits the consistency checking of NFRs with overlapping scopes.

We introduce and compare three alternative strategies in the case that multiple NFRs exist and analyze their relative advantages and disadvantages under different scenarios. This approach to verifying the NFRs can also support efficient re-verification of composite services as needed when NFRs are updated.

The rest of the paper is organized as following. Section II provides an overview of our approach. Section III briefly reviews an approach to exploring the space of candidate compositions. Section IV describes our approach to modeling NFRs and several strategies for verifying whether a candidate composition satisfies the NFRs. Section V describes related work. Section VI concludes with a brief discussion and limitations and directions for further research.

II. Overview

Our overall approach to assembling a composite service that satisfies both the functional and non-functional requirements (in the case of security related NFRs) is shown in Fig. 2. The functional composition algorithm [9] (briefly reviewed in Section III below) uses a goal model in the form of an automaton to encode the user-specified FR. A composite service is represented as an automaton where states represent the component services participating in the composition and inter-state transitions represent composition of the corresponding component services. The functional composition algorithm assembles a composite service that verifiably satisfies the FR by exploring the space of candidate compositions.

The focus of this paper, however, is on verifying that a composite service assembled by the functional composition algorithm *also* satisfies the NFR. We use an automaton to represent the non-functional properties of a composite service (See Section IV for details). The latter correspond to sequences of properties of non-functional attributes of the component services that make up a composite service. Labels on transitions of the NFR automata encode the set of non-functional *constraints* that are satisfied by the services that are connected by the respective transitions.

A composite service is said to satisfy a given set of NFRs if and only if the sequence of properties over non-functional constraints represented by the NFR-automata is also realized by the automaton that encodes the composite service. A composite service conforming to a desired NFR is obtained by computing and verifying the product of the automaton representing the composite web service and the automaton representing the corresponding NFR. This lifts the NFR analysis from the level of individual services to the level of the search space of candidate compositions obtained from the functional requirements.

The non-functional properties, e.g., security policies, of the component services are extracted from their WSDL specifications. A domain terminology mapping table is used to translate the user-specified NFR, e.g., "messages shall be highly-encrypted" into the terminology used in the WSDL specification ("messages must use standard Basic128Sha256 encryption algorithm with a 256-bit key").

The verification of a composite web service consists of two main parts, as shown in Fig. 2. The first part is the functional composition of the component web services. This is prior work [9], briefly reviewed in Section III below. The goal model is an automaton constructed to model the functionalities of the required system. With the goal model as input, an iterative algorithm generates a composite web service that verifiably satisfies the functional requirements.

The second part of the process, and the contribution of this paper, is the verification of the security non-functional requirements, shown in the bottom half of the diagram. The security NFR is modeled in an automaton-based model, as described in Section IV below. The composite web service derived from the first step can then be verified against this security automaton. During the verification, the security policies of the service operations are pulled from the appropriate WSDL files. The domain terminology mapping table is used to translate the security policies into an algorithm-readable input and to provide the algorithm with the aggregation rules needed to handle security NFRs.

It must be noted that the focus of this paper is on static verification of NFR. Verification of NFRs where satisfaction is contingent on runtime performance (e.g., whether an ambulance reaches its destination in time to meet a response time NFR) can only be dynamically verified at runtime and are beyond the scope of this paper.

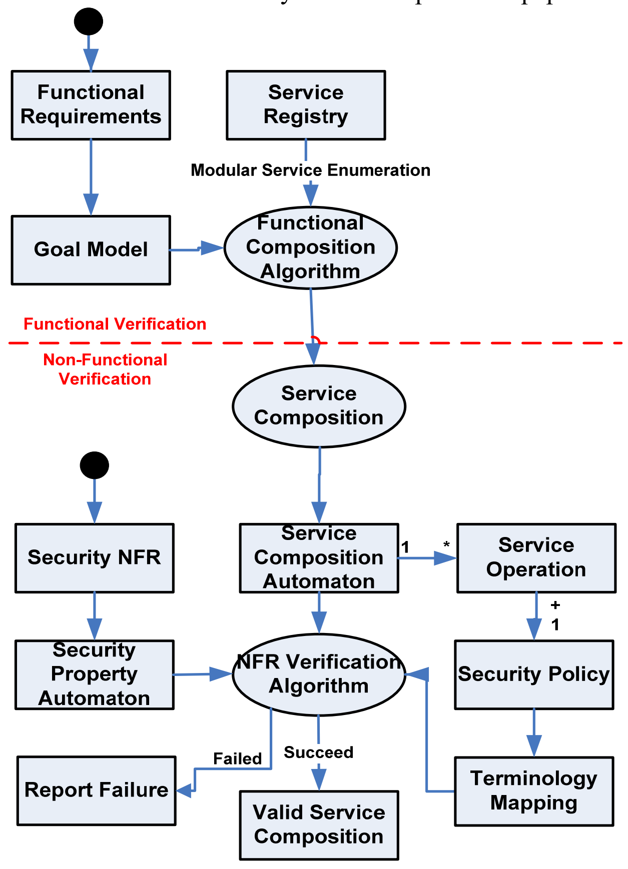

Functional Requirements
Service Registry
Modular Service Enumeration
Goal Model
Functional Composition Algorithm
Functional Verification
Non-Functional Verification
Service Composition
Security NFR
Service Composition Automaton
1
*
Service Operation
+
1
Security Property Automaton
NFR Verification Algorithm
Security Policy
Failed
Succeed
Report Failure
Valid Service Composition
Terminology Mapping

Figure 2. Overview

III. FUNCTIONAL COMPOSITION

In our approach, we take the pre-construction of the composite web service meeting the FRs as fundamental. This is illustrated in the top half of the overview in Fig. 2. In our illustration, we use the algorithm described in previous work [9] to achieve this construction, but other approaches, such as those in [1, 2], would also work.

The goal model in Fig. 3 is constructed as a first step in composing the component services into a composite service. The goal model captures two types of information required by the FRs: the workflow of the composite service and the component level functionality for each service operation. The states represent component services and the transitions represent the input/output messages for the service operations. Note that one component service can have more than one operation, so may be represented by multiple states. For example, the ambulance service can have service operations to provide availability information regarding ambulance resources and to reserve these resources.

The label on each transition represents a guard condition or service operation functionality required to be realized by a component service, such as Scheduler_Reserve. Each operation can take message inputs and produce message outputs. The goal model identifies the service operations and workflows of the EMS as a functionality template for required EMS candidates. The functionalities, such as scheduling, reserving and dispatching firetrucks, ambulances, and police are specified in the goal model.

The goal of the functional service composition algorithm is to find component services providing the required functionalities. The Iterative Forward and Backward Search Algorithm [9] is invoked with the goal model in Fig. 3 as input to generate the composite web service. The algorithm maintains a composition result set during the computation and tentatively searches forwards and backwards in the component service registry for a service that satisfies the functional requirement. The result set moves forward if the current component service satisfies the service operation specified in the goal model and stores it in the result set; otherwise, it moves backwards and tries another component service. The algorithm uses an exhaustive search to find all composite web services satisfying the goal model finally.

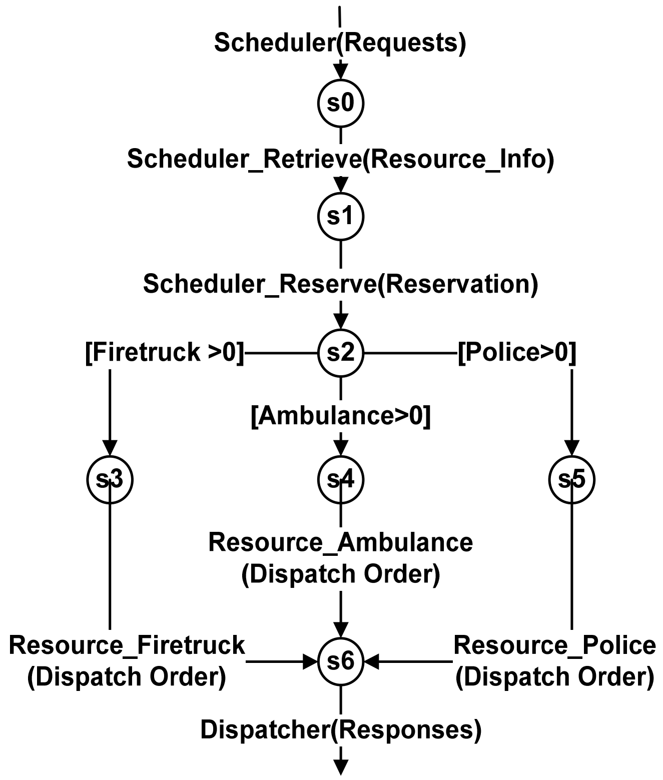


Figure 3. Functional goal automaton of the EMS

IV. SECURITY REQUIREMENT VERIFICATION

To be able to verify the NFRs, the first step is to model them. As shown in the overview in Fig. 2, each security NFR is modeled as a security property automaton.

A. NFR Automata Derivation

We thus first define the finite state automaton used to model the composite web service and the NFRs.

Definition 1: A finite state automaton is a tuple FSA = (S, s_0, Δ, P, F) where S is the finite set of states, $s_0 \in S$ is the start state, and $\Delta \subseteq S \times 2^P \times S$ is the transition relation of the form $s - \phi \rightarrow s'$ such that $s, s' \in S$, and $\phi \in 2^P$ is a subset of propositions P. Finally, $F \subseteq S$ is the set of final states.

A finite sequence is said to be accepted by the FSA if and only if the sequence starts from s_0 and terminates in any of the final states in F. The NFRs and composite web services are described using FSA.

Fig. 4 shows a composite web service automaton derived from the FR goal model in Fig. 3. This candidate represents an instance of the EMS where only police need to be dispatched. The composite web service is here augmented with its associated security NFRs on the transitions.

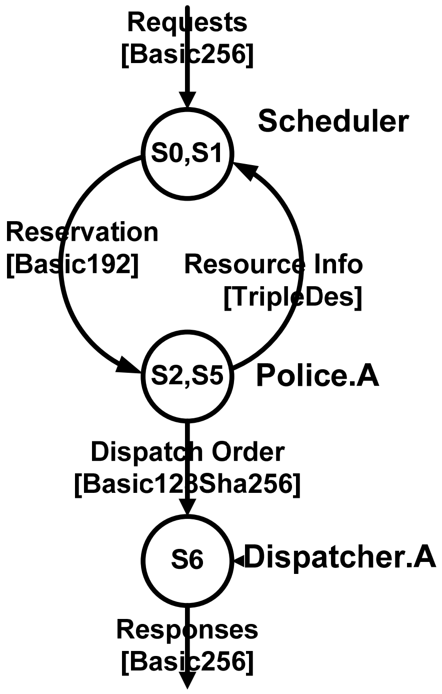


Figure 4. A composite web service candidate automaton of the EMS with security policies associated

We will also model the NFR property constraints using the automaton in Definition 1. Note that an automata-based property model can represent both safety and liveness properties [4, 5], where a safety property is of the form "a program never enters an undesirable state" and a liveness property is of the form "a program eventually enters a desirable state". If a state violating a safety property is encountered during composition of the composite web service automaton and the NFR automaton, or if a state satisfying a liveness property is not reached in any branch at the end of the composition, we say that this composite web service violates the NFR. To achieve compositional verification of the multiple NFR properties, we unify the types of properties by converting the liveness properties to safety properties via use of an additional trap state π to capture those undesirable final states in the liveness properties. This is described more fully in [4].

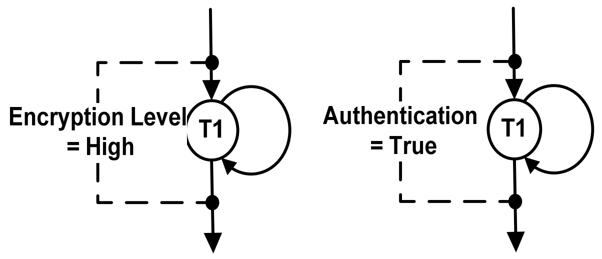


Figure 5. Global security requirements: (1) all service operations shall employ highly-secured messages; (2) all service operations shall have authentication.

Scoping the NFRs. The scope of each NFR constraint is specified as the dotted line as shown in the excerpts in Fig. 5. A global NFR property can be described as a single self-loop with a NFR property constraint on it. For example, Fig. 5 shows two global security properties. One requires that all service operations shall employ a high encryption level and the other requires that all messages and operations shall be authenticated. The concept of "Encryption Level = High" and "Authentication = True" in the constraints will be defined in the domain terminology mapping table in Section IV.B.

In the composite web service, a NFR may refer only to the properties or behaviors of some services or some message paths. A *scope* of a NFR property refers to a user-defined subset of the services or service operations to which the NFR property actually applies. For example, a user of the EMS for a search and rescue incident may require only the service operations of reading and writing the Police Resource to be highly encrypted, rather than globally requiring all service operations to be highly encrypted. Similarly, that user may require only the message to dispatch the police to be authenticated. In the excerpts in Fig. 6, the dotted lines describe the local scopes of these security properties.

The FR goal model introduced in Section III is used here as a template to specify NFR constraints and their scoping information. Since all composite web service candidates have the same workflow as that specified in the goal model, a trace equivalence check can be performed to verify that the composite web services satisfying the FRs also satisfy the required security constraints.

We now derive the security NFR constraint model from the goal model in Fig. 3. We illustrate the process with the NFR described in Section I, i.e., that a request to dispatch police shall be processed using highly-encrypted message paths. The construction of a NFR property constraint consists of the following steps:

1. Identify the scope in the goal automaton (Fig. 3) for this NFR constraint. For the security NFR of concern here, we identify the path with the Police_Resource operation in it to be the scope.
2. Label the NFR constraint for the scope to cover all the operations within the scope. Here we label the path we identified in step 1 as "Encryption Level>=High".
3. Simplify the model, if possible, by merging the states and transitions unrelated to the scope into a single state with a self-loop.
4. Prune unrelated states and paths. Here we remove the other branches from the graph, yielding the security NFR automata in Fig. 6.

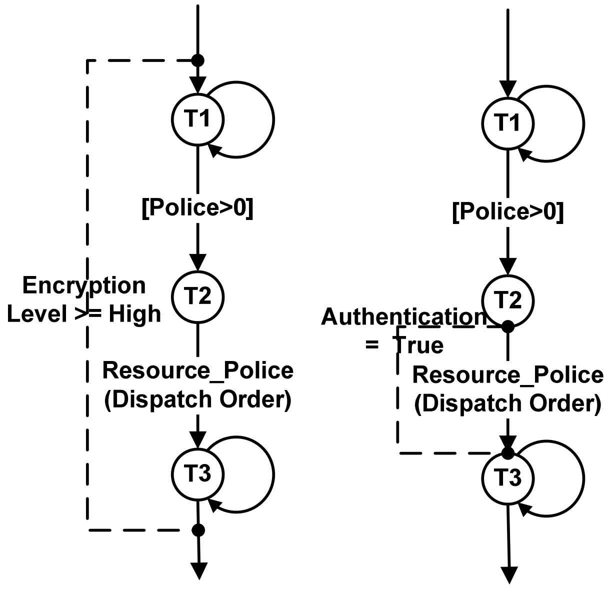


Figure 6. Locally scoped security requirements: (1) all requests to dispatch police shall employ highly-encrypted messages; (2) messages to dispatch the police shall be authenticated.

To verify that the composite web service satisfies the security NFR, we must first retrieve the security guarantees of the individual component services. The composite web service produced by the functional composition, e.g., Fig. 4, is stored in Web Service Business Process Execution Language (WSBPEL) [19] file. This WSBPEL file records the workflow of the participating component services and associates with their WSDL files, which further describes their service operations. Each service operation, as shown in Fig. 4, is associated with a set of NFR attributes in the WSDL file, namely the security property described in the WS-Security. The security policies named in Fig. 4, such as Basic192, are different security policies which each contain the description of an encryption algorithm, a signature key length, a symmetric or asymmetric key and other security attributes. The following XML excerpt shows how a security policy is associated with a service operation:

Security Policy Binding to Service Operations

```
<Policy wsu:Id="medium_secure">
 <ExactlyOne>
```

```
    <sp:Basic192 ... />
      </ExactlyOne>
    </Policy>

    <wsdl:binding name="SecureBinding"
type="tns:ReservationInterface" >
      <PolicyReference URI="# medium_secure " />
      <wsdl:operation name="Reservation"
>...</wsdl:operation>
      ...
    </wsdl:binding>
```

In the first part of this piece of code, a Basic192 security solution (described in [22]) is defined in the security policy. In the second part, this policy is bound with the Reservation operation in the WSDL file of the police service. This reservation operation reserves the police resources to dispatch and prevents concurrent allocation of the same units. The binding between the security policy and the service operation allows our approach to retrieve the security policy for each operation in a composite web service for verification.

As shown in Fig. 4 and the XML code, the security policies are specified for each operation of each component of a composite service. Note that the security policies of different operations can differ from each other. The resulting sets of security policies (and the associated enforcement algorithms) are collectively used to construct a domain terminology mapping table that allows the different security policies to be compared with each other.

B. Domain Terminology Mapping

During verification, the security policies of the service operations associated with a service are retrieved from the WSDL specifications of the corresponding service. A domain terminology mapping table is designed to bridge the gap between the user's requirements (Low, Medium, High, or Critical level encryption) and the specifications described in the web services (algorithms and minimum key lengths). Table I defines, for our illustrative example, an excerpt of the different encryption levels specified in WS-Policy [22] and a Boolean option for an authentication feature for the composite web service. These choices associate with different security policies in WS-Security used by the web services.

The security solutions specified in WS-Security are mapped to an encryption level based on the minimum key length of their encryption algorithms. By using a symmetric encryption algorithm, the message sender and the receiver can share a key, in order that only the genuine sender can encrypt and the real receiver decrypt the communication [31]. The optional authentication feature is mapped to the security policies based on whether a symmetric encryption is applied. These mappings are stored in a table for use and reuse in verification that candidate composite services satisfy the security NFRs.

TABLE I. SECURITY TERMINOLOGY MAPPING TABLE EXCERPT

Encryption Level	Security Algorithm	Min Key Length	Sym-metric
Low	None	0	No
Medium	Basic128	128	No
	Basic128Sha256	128	Yes
High	Basic192	192	No
	Basic192Sha256	192	Yes
	TripleDes	192	No
	TripleDesSha256	192	Yes
Critical	Basic256	256	No
	Basic256Sha256	256	Yes

Authentication	Security Algorithm	Min Key Length	Sym-metric
False	None	0	No
	Basic128	128	No
	Basic192	192	No
	TripleDes	192	No
	Basic256	256	No
True	Basic128Sha256	128	Yes
	Basic192Sha256	192	Yes
	TripleDesSha256	192	Yes
	Basic256Sha256	256	Yes

C. NFR Aggregation Rules

An aggregation rule combines the valuations of non-functional attributes of component services that fall within the scope of the corresponding NFR [11]. For example, in the case of the EMS encryption requirement, the aggregation rule is defined as: the encryption level of the composite service is the minimum encryption level of the component services that fall within the scope of the NFR for EMS encryption. We implement this aggregation rule in the following aggregation script, which is used by the security NFR verification algorithm.

Encryption Level Aggregation Script:

```
1:  Define Low=0
2:  Define Medium=1
3:  Define High=2
4:  Define Critical=3
5:  Define Initial_Security_Level=3
6:  Int Aggregate(Int current_security, Int
    new_security)
7:  {if(current_security>new_security){
8:  current_security=new_security; }
9:  return current_security;
10: }
11: Bool isSatisfied(Int current_security, Int
    desired_security)
12: {
13: If(current_security< desired_security) return false;
14: return true;
15: }
```

D. NFR Verification Algorithm

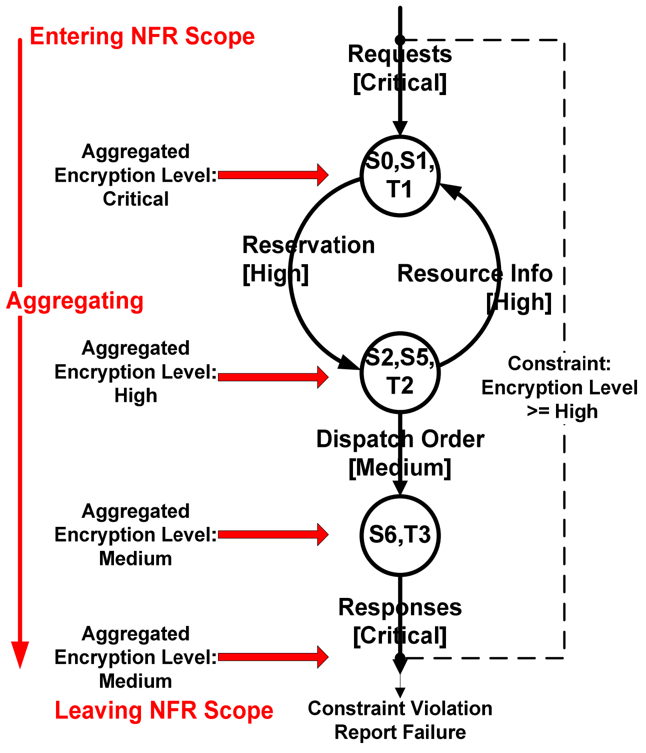
In order to verify that a NFR is satisfied in a candidate composite web service known to satisfy the FRs, we compose the automata representation of the composite web service and the NFR property. In our approach, all automata compositions are synchronous, i.e., multiple automata can make progress in parallel for each step [15]. The problem of whether a composite web service conforms to a desired NFR is addressed by calculating the synchronous product of automata representing the composite web service with those representing the corresponding NFRs. The composite web service is said to satisfy the NFRs if and only if the sequence of properties over non-functional attributes as represented by the NFR-automata is also present in the automaton of the composite web service. The verification process can be viewed as an equivalence check.

Definition 2: Given two automata, $FSA_i = (S_i, s_{0i}, \Delta_i, P_i, F_i)$, for $i \in \{1, 2\}$, their product is another FSA denoted by $FSA_1 \times FSA_2 = (S_{12}, s_{012}, \Delta_{12}, P_{12}, F_{12})$, where $S_{12} \subseteq S_1 \times S_2$, $s_{012} = (s_{01}, s_{02})$, $P_{12} = P_1 \cup P_2$, $F_{12} = \{(s_1, s_2) \mid s_1 \in F_1, s_2 \in F_2\}$. Finally, $s_1 - \phi_1 \rightarrow s_1' \in \Delta_1$ and $s_2 - \phi_2 \rightarrow s_2' \in \Delta_2$ and $(s_1, s_2) - \phi_1 \wedge \phi_2 \rightarrow (s_1', s_2') \in \Delta_{12}$.

The automata composition concept can be used in two ways: (1) to verify a NFR automaton with a composite web service, and (2) to verify the consistency of two or more NFR automata if two or more NFR properties need to be combined.

When the automata composition algorithm encounters any service within the scope, then, if there is an aggregation rule associated with that NFR, the aggregation script that implements the aggregation rule is included in the verification algorithm.

The verification algorithm uses an entrance condition to detect entry into the relevant scope for the current NFR, and an exit condition to trigger evaluation of whether the current NFR has been satisfied in the composite web service. Fig. 7 shows the composed automaton and how aggregation is applied to the scope range for the security NFR in the EMS example during automata composition. During the composition of the composite web service automaton and the security property automaton, the states and transitions of both automata merge when they share the same predicates. When the composition enters the scope of the security constraint, the encryption level value starts to aggregate for each merged transition according to its aggregation rule and the terminology mapping table. When the composition algorithm leaves the scope, the aggregated encryption level value is compared to the value required by the constraint to evaluate its satisfaction (here, a High encryption level). All these tasks are described inside the aggregation script for the implementation.



Entering NFR Scope
Requests [Critical]
Aggregated Encryption Level: Critical
S0,S1, T1
Reservation [High]
Resource Info [High]
Aggregating
Aggregated Encryption Level: High
S2,S5, T2
Constraint: Encryption Level >= High
Dispatch Order [Medium]
Aggregated Encryption Level: Medium
S6,T3
Responses [Critical]
Aggregated Encryption Level: Medium
Leaving NFR Scope
Constraint Violation Report Failure

Figure 7. Security property aggregation and composed automata

The algorithm to verify a NFR property automaton against a composite web service automaton is described in pseudo code as follows:

Security Level Verification Algorithm:

```
Let $FSA_c = (S_c, s_{0c}, \Delta_c, P_c, F_c)$ be the composite web service automaton.
Let $FSA_n = (S_n, s_{0n}, \Delta_n, P_n, F_n)$ be the security constraint automaton.
Let $FSA_p = \emptyset$ be the initial composed automata.
1:  Load Security Level Aggregation Script
2:  Set current_security= Initial_Security_Level
3:  Initial state of $FSA_p$ : $s_{0p} = \{s_{0c}, s_{0n}\}$
4:  Call Combine_States($s_{0c}, s_{0n}, s_{0p}$ )
5:  If isSatisfied(current_security, required_security) report
    success, else report failure.
6:  proc Combine_States($s_{0c}, s_{0n}, s_{0p}$){
7:  /*Select the states with satisfied guard conditions */
8:   ForEach $\phi_c \in 2^{Pc}, s_c' \in S_c$ s.t. $\forall s_n' \in S_n$:    $s_{0n}$
    $- \phi_n \rightarrow s_n'$ , $s_{0c} - \phi_c \rightarrow s_c'$ and $\phi_c \subseteq \phi_n$
9:  Create a new state $s_p' = \{s_c', s_n'\}$ for $FSA_p$
10: If $s_n'$ is a trap state, return and report failure.
11: Create a new transition $s_{0p} - \phi_c \rightarrow s_p'$ for $FSA_p$
12: If InScope,
13:        If the guard condition $\phi_c$ is a service operation,
           retrieve its security policy name as SName from its
           WSDL file.
14:     Retrieve the security level as new_level by
        searching for SName in the security terminology
        mapping table.
15:     current_security=Aggregate(current_security,
        new_level)
16: Call Combine_States($s'_c$, $s'_n$, $s'_p$)
17: End ForEach
18: }
```

Note that this algorithm is for static verification, while for dynamic verification (using runtime data), additional termination criteria are needed.

A verification of the security property in Fig. 7 on the composite web service in Fig. 4 returns a "Failure" result, because the aggregation result of the encryption level for this composite web service candidate is Medium in its required scope, which is lower than the required High.

E. Handling Multiple Properties

In complex composite web services, there often exists more than one user specified non-functional requirement to verify, such as security or availability constraints with different scopes. An interesting issue is how to handle the multiple property automata efficiently. Given a candidate composite web service from the search space, the most straight-forward way to verify each property automata is to verify the properties independently and sequentially. This strategy, Independent Composition (INC), is shown in the first column in Table II. INC calculates each property independently with the candidate composite web service and discards the intermediate calculation results after each inner loop. INC returns verification failed at the first unsatisfied NFR property. It is most suited to a search space with few candidates. A disadvantage of INC is that even if the properties are inconsistent, such that no composite web service can satisfy them all, the algorithm has to explore the entire search space before telling the user that no composite web service satisfies the NFR.

To overcome this weakness of the INC algorithm, we introduce the Two-Stage Composition (TSC) algorithm (the second column of Table II) which detects property inconsistency before performing the verification. TSC composes all property automata into one combined property automaton prior to verification. In this way, the first inconsistency found during property composition will terminate the verification process, and the property automaton causing this failure will be captured and returned to the user for possible modification of the NFR. In addition, TSC can assist with efficient verification, since property automata may share the same predicates on the transitions. The more predicates shared by the property automata, the fewer states the combined automaton will have. Another advantage of using TSC is that the combined property automata can be reused to verify multiple candidate composite web services or when NFRs are updated.

A third strategy, the Big-Bang Composition (BBC) algorithm, replaces the sequential verification of the properties in INC with parallel verification (the third column in Table II). BBC composes all the automata including the candidate composite web service and the property automata synchronously so that the earliest reachable trap state in any of the properties can terminate the verification by returning a failure.

A combined strategy can be applied using a smart switcher, which pre-calculates the verification overhead when provided a group of composite web service candidates and a group of NFR properties. Generally speaking, INC and BBC are useful for detecting inconsistencies in the candidate composite web services quickly, whereas TSC is most useful in quickly locating inconsistencies in the user-defined NFRs.

V. Related Work

Most existing composite web service verification approaches focus on satisfying the functional requirements [1, 2]. However, several researchers have also described NFRs in the context of Quality of Service (QoS) [10, 11, 12].

Research on modeling and verifying *hard* constraints follows three main approaches: context-matching based techniques [12, 13], axiom-based techniques [8, 11, 14] and automata based techniques [7, 9, 10].

Context-based property models can be verified by context-based matching, including syntactic matching and semantic matching, in order to ensure compliance of the composition to the functional and non-functional requirements [12, 13]. The drawbacks for context-based matching are limited efficiency of searches and inaccuracies in key word identification.

Rao, Kungas and Matskin introduce an axiom-based method for semantic web service composition using Linear Logic theorem proving [8]. Zeng et al. [11] introduce optimizing preferences over NFRs based on utility functions in linear programming. However, the Linear Logic prover requires expert knowledge to construct property models. Temporal logic-based property models are commonly used in planning and model checking based web service composition [2]. These approaches require expert knowledge and pre-construction of a formal model for each candidate composite web service. They also can face the state explosion problem during verification and the problem of consistency assurance among the property models.

Modeling NFRs using automata allows one to take advantage of existing automata-based approaches to functional verification. Foster, Uchitel, Magee and Kramer describe a model-based approach (LTSA-WS) to verify composition implementations on functional properties [7]. Pathak, Basu and Honavar [9] introduce an automata-based goal model to represent the desired functionality of a composite service and describe and an algorithm for assembling a set of component services to obtain a composite service that achieves the desired functionality. However, existing automata-based approaches focus primarily on the functional aspects of composition.

There is a growing interest in techniques for composite web services that take into account both functional and user preferences over non-functional attributes of a composition. Such preferences can be quantitative or qualitative in nature.

TABLE II. THREE STRATEGIES FOR VERIFYING MULTIPLE PROPERTIES

	Independent Composition	Two-Stage Composition	Big-Bang Composition
Pseudo-Code	1. For each candidate composition s_i in S 1) For each property p_j in P 2) Verify this property by calculating $s_i \times p_j$ 3) If the verification failed, jump to the next s_i 4) End For Each 2. Output s_i and terminate 3. End For Each 4. Output "No composition satisfies the NFR"	1. Take automata $q = p_0$ 2. For j = 1 to k 3. Calculate $q \times p_j$ and Save the result as q 4. If there exists only one termination state in q and it's a trap state in the safety property, Output "Property p_j is Inconsistent with the other properties" and Terminate 5. End For 6. For each candidate composition s_i in S 7. Verify the property by calculating $s_i \times q$ 8. If the verification failed, jump to the next s_i 9. Else Output s_i and terminate 10. End For Each 11. Output "No composition satisfies the NFR	1. For each candidate composition s_i in S 2. Verify the property by calculating $s_i \times p_1 \times p_2 \times \ldots \times p_n$. All automata are composed synchronously rather than pair-wisely. When a trap state is reached, the composition terminates and returns failure. 3. If the verification failed, try the next s_i 4. Else Output s_i and terminate 5. End For Each 6. Output "No composition satisfies the NFR"
Com-plexity	$O(\sum_{i=1}^{n}\sum_{j=1}^{k}(\lvert s_i\rvert \times \lvert p_j\rvert))$	$O(\sum_{i=1}^{n}(\lvert s_i\rvert \times (\prod_{j=1}^{k}\lvert p_j\rvert)))$	$O(\sum_{i=1}^{n}(\lvert s_i\rvert \times \prod_{j=1}^{k}\lvert p_j\rvert))$
S is the set of all compositions. S = $\{s_1, s_2, s_3, \ldots. s_n\}$ n is the number of compositions in the search space P is the set of all NFRs. P = $\{p_1, p_2, p_3, \ldots. p_n\}$ k is the number of NFRs			

Quantitative preferences are expressed using cost functions or utility functions to be optimized by the composition algorithm [17]. Qualitative preferences are modeled using preference networks, conditional preference networks, or their variants [11, 16, 18].

There is a body of work that focuses on the verification of web service reliability during service composition. Foster [32] introduces a service behavior model to ensure that a composite service is free of deadlocks. Techniques for assessing the reliability of composite services rely on models of faults and failures [21, 23, 33]. In other work, Mikalsen, Rouvellou and Tai propose a model of web transactions to improve the reliability of composite web services [24].

In web service security, Raya et al. identify vulnerabilities including jamming, forgery, in-transit tampering, impersonation, privacy violation and on-broad tampering [26]. Weiss and Mouratidis model security requirements using patterns, and global security requirements of a composite service are verified using pattern matching [30]. Local scoping and inconsistency are not addressed.

VI. CONCLUSION

This paper shows how an automatically generated composite web service of independently developed web services can be verified to meet the non-functional security requirements imposed by the user as hard constraints. The approach described here enables this verification by lifting the NFR analysis from the level of individual services to the level of the search space of candidate composite web services obtained from the functional requirements. The primary limitations of this approach are (1) that it currently can only handle those types of NFRs which can be specified in WSDL and WSDL's auxiliary specifications, such as WS-Policy, WS-Security, WS-Trust and WSLA, and (2) that domain expert knowledge is needed to build the terminology mapping table. We hope to ease these restrictions in future work.

ACKNOWLEDGMENT

This research is supported by NSF grants 0541163, 0702758 and 0916275, the latter with funds from the American Recovery and Reinvestment Act of 2009. We thank the reviewers for several helpful suggestions.

REFERENCES

[1] S. Dustdar and W. Schreiner, "A Survey on Web Services Composition," Int'l Journal of Web and Grid Services, vol. 1, No.1, 2005, pp. 1–30.

[2] N. Milanovic and M. Malek, "Current Solutions for Web Service Composition," Internet Computing, Nov/Dec 2004, vol. 8, No. 6, pp. 51-59.

[3] L. Chung. B. A. Nixon, E. Yu and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Springer, 1999.

[4] S. Cheung and J. Kramer, "Checking Safety Properties Using Compositional Reachability Analysis," ACM TOSEM, vol. 8, No. 1, Jan 1999, pp. 49-78.

[5] H. Guo, Y. Shin and W. Lee, "Enhanced Compositional Safety Analysis for Distributed Embedded Systems using LTS Equivalence," Proc. 6th ICACS, vol. 6, 2007, pp. 115-120.

[6] B. Bruegge and A.H. Dutoit, Object-oriented Software Engineering: Using UML, Patterns and Java, Prentice Hall, 2003, pp. 181-196.

[7] H. Foster, S. Uchitel, J. Magee and J.Kramer, "Model-based Verification of Web Service Compositions," ASE, 2003, pp. 152-163.

[8] J. Rao, P. Kungas, and M. Matskin, "Logic-Based Web Services Composition: from Service Description to Process Model," ICWS, 2004, pp. 446-453.

[9] J. Pathak, S. Basu and V. Honavar, "Modeling Web Services by Iterative Reformulation of Functional and Non-Functional Requirements," 4th ICSOC, 2006, pp. 314-326.

[10] M. C. Jaeger, G. Rojec-Goldmann and G. Muhl, "QoS Aggregation for Web Service Composition using Workflow Patterns," Proceedings of the Enterprise Distributed Object Computing Conference, 2004, pp. 149–159.

[11] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam and Q.Z. Sheng, "Quality Driven Web Services Composition," Proc. 12th Int'l Conference on World Wide Web, 2003, pp. 411–421.

[12] K. Zachos and N. Maiden, "Inventing Requirements from Software: An Empirical Investigation with Web Services," RE '08. 2008, pp. 145-154.

[13] B. Medjahed and Y. Atif, "Context-Based Matching for Web Service Composition," Distributed and Parallel Databases vol. 21, No. 1, 2007, pp. 5-37.

[14] Pistore, F. Barbon, P. Bertoli, D. Shaparau and P. Traverso, "Planning and Monitoring Web Service Composition," Artificial Intelligence: Methodology, Systems, and Applications, Springer Berlin / Heidelberg, 2004, pp. 106-115.

[15] M. Huth and M. Ryan, Logic in Computer Science, Cambridge University Press, 2004.

[16] G. Santhanam, S. Basu and V. Honavar, "TCP - Compose * --- A TCP-Net Based Algorithm for Efficient Composition of Web Services Using Qualitative Preferences," Proc. 6th ICSOC, 2008, pp. 453-467.

[17] S. Sohrabi, J. Baier and S. McIlraith, "HTN Planning with Quantitative Preferences via Heuristic Search," Oversubscribed Planning and Scheduling Workshop of ICAPS, Australia, 2008.

[18] S. Sohrabi, N. Prokoshyna and S. McIlraith, "Web Service Composition via Generic Procedures and Customizing User Preferences," 5th ISWC, 2006, pp. 597-611.

[19] OASIS, OASIS Web Services Business Process Execution Language (WSBPEL) TC, Available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[20] W3C organization, "Web Service Policy," Available at http://www.w3.org/Submission/WS-Policy/

[21] L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan and D. Theseider-Dupre, "Advanced Fault Analysis in Web Service Composition," 14th International Conference on World Wide Web, 2005, pp.1090-1091.

[22] S. Bajaj, et. al. , "Web Services Security Policy Language Specification V1.2," W3C member submission, 2006, pp. 30-34. Available at http://www.w3.org/Submission/WS-Policy/

[23] D. Zhang, Z. Qi and X. Xu, "Reliability Prediction and Sensitivity Analysis of Web Services Composition," Petri Net, Theory and Applications, I-Tech Education and Publishing, 2008. pp. 20-31.

[24] T. Mikalsen, I. Rouvellou, and S. Tai, "Reliability of Composed Web Services--From Object Transactions to Web Transaction," Workshop on Object-Oriented Web Services, OOPSLA 2001, Tampa, Florida, 2001.

[25] A. Lamsweerde, "Requirements Engineering: From System Goals to UML Models to Software Specifications," Wiley, 2009, pp.24.

[26] M. Raya, P. Papadimitratos and J. Hubaux, "Securing Vehicular Communications," Wireless Communications, IEEE Publication, Oct 2006 vol. 13, Issue 5, pp. 8-15.

[27] A. Avizienis, J.C. Laprie, B. Randell and C.E. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Trans. Dependable Sec. Comput. 1(1), 2004, pp. 11-33.

[28] P. Neumann, "The Risk Digest: Forum On Risks To The Public In Computers And Related Systems," ACM Committee on Computers and Public Policy, vol. 4, Issue 52, Available at: http://catless.ncl.ac.uk/Risks/

[29] B. O'Neill, "A Model Assessment Tool for the Incident Command System: A Case Study of the San Antonio Fire Department," Applied Research Projects, Paper 270, 2008.

[30] M. Weiss and H. Mouratidis, "Selecting Security Patterns that Fulfill Security Requirements," Proc. of RE08, pp. 169-172.

[31] W. Stallings, Network Security Essentials: Applications and Standards, Prentice Hall, 3rd Edition, 2007.

[32] H. Foster, "Tool Support for Safety Analysis of Service Composition and Deployment Models," ICWS 08, 2008, pp.716-723.

[33] J. Wu and F. Yang, "QoS Prediction for Composite Web Serviceswith Transactions," LNCS, Springer Berlin/Heidelberg, vol. 4652, 2007. pp. 86-94.