Policy based Web Service Orchestration and Goal Reachability Analysis using MSC and CP Nets.

Aabhas V. Paliwal  
Rutgers University,  
CIMIC  
aabhas@cimic.rutgers.edu

Nabil Adam  
Rutgers University,  
CIMIC  
adam@cimic.rutgers.edu

Basit Shafiq  
Rutgers University,  
CIMIC  
basit@cimic.rutgers.edu

Christof Bornhoevd  
SAP Labs, LLC  
christof.bornhoevd@sap.com

Abstract

Specification of a web services-based process (WSP) for a complex environment such as homeland security domain is a complex task. The specification task requires achieving a balance between intuitive and easy to understand process representation for the interactive domain user and consolidated formal specification. An easy to understand process representation facilitates conformance of process correctness where as the formal specification ensures formal goal reachability analysis of service orchestration for executing the process. Existing approaches provide means for service orchestration and invocation, however, there is little to support the specification and goal reachability analysis of WSPs representing distributed service oriented architectures. In this paper we present a novel approach to service orchestration that combines an effective diagrammatic modeling, an appropriate formal framework and an implementation process for dynamic WSP and complex web service composition and goal reachability analysis. Specifically, our approach for WSP specification and goal reachability analysis comprising of service orchestration using High Level Message Sequence Charts (HMSC) and Colored Petri Nets (CP Nets) provides a methodology for instantiation of the general WSP that is specific to individual users. The process instantiation takes in to account the dynamicity aspects in terms of the changes in the domain requirements, the user request and the component services that form the WSP.

1. Introduction

A large number of web services are being developed as independent software building blocks to construct distributed applications as web services-based processes (WSPs). WSPs are similar to enterprise business processes and comprise of web services orchestrated together into a logical flow for a resulting configuration of services to behave as a cohesive application [7]. Web services that make up the WSP represent semantically characterized functionality that can be discovered or semi-automated or interactively composed to deliver advanced domain functionality. This advanced functionality may involve WSPs spanning multiple organizations that integrate services and resources as illustrated by our motivating scenario in the Homeland Security domain.

Orchestration of services involves overall service specification, service selection, and composition of services to achieve the overall goal. Goal reachability facilitates analysis of WSP output based on user constraints, before the actual execution of the composed process, given the complex interaction between the participant services.

Specification of a WSP for a complex environment such as the homeland security is a complex task that needs to be supported by easy to understand diagrammatic and appropriate formal specification techniques. Existing approaches (e.g., [9, 17]) only provide means for service composition and invocation, however, they offer little support for the specification and goal reachability analysis of WSPs representing distributed service oriented architectures (SOA) [8]. Other existing approaches (e.g., [3, 14, 18]) do not provide a formal representation for service composition and goal reachability analysis. There is a need for a methodology that achieves WSP specification with: 1) diagrammatic representation of the domain activities, 2) automatic transformation into a formal specification and 3) execution using programming constructs. The diagrammatic representation facilitates process goal reachability analysis at a higher level of orchestration and provides a visual check of the correctness of the process. The formal specification facilitates goal reachability at service composition level. The general idea of goal reachability analysis is to enumerate the set of reachable states for the program under consideration. In our case goal reachability analysis provides a simulation of the output provided to the requestor by a WSP. The output generated by the WSP is based on the component services invoked, that are determined by user credentials and characteristics. Existing approaches, (e.g., [20, 19, 1]), fall short in achieving a balance between intuitive and easy to understand representation for the interactive user, and consolidating formal specification for the (semi-) automatic validation.

In this paper we propose a new approach for WSP specification and goal reachability analysis, comprising of service composition using HMSCs and Colored Petri Nets.
Specifically, our contribution provides a methodology for precise and coherent specification at a process level and goal reachability analysis, based on policy constraints, at the service level.

We utilize HMSCs for the specification of the WSP. Next, we employ CP Nets to provide an appropriate formal framework and effective modeling components for dynamic and complex web service analysis and synthesis. The basic underlying idea is to specify service behavior in a process by a HMSC. Next we propose to map the component MSC to CP Nets and propose to use CP Nets for goal reachability analysis of web service composition, relying on well-known Petri Nets concepts, i.e., reachability analysis. This is followed by transforming the CP Nets into a BPEL process that provides block structured language for implementation.

The paper is organized as follows, before we lay out the details of our approach in section 3, we first discuss the preliminaries in section 2. Related work is presented in section 4. Finally, our conclusions and future work are described in section 5.

1.1 Motivating Scenario

Our motivating scenario assumes a radiological threat event which is responded by a Virtual Multi-Agency Response Team Emergency Management System (VMART EMS) [2] as illustrated in Figure 1. The first instance of information sharing and collaboration involves the coherent integration of operational protocols from multiple agencies covering local, state, and federal jurisdictions and employing varied resources at their disposal. One of the ways for this to be accomplished is via an integrated system of systems utilizing assets at the local, state and federal levels. The integrated system is based on a service-oriented architecture wherein the participating agencies dispense their responsibilities utilizing functionality provided by local as well as third party web services. The different types of services involved are depicted in Figure 1.

The scenario unfolds as follows, once a sensor is triggered, a secondary inspection will be performed. If the second sensor identifies that the source could be a threat, an alert is generated and sent to the local emergency operations center. If the second sensor cannot identify the source, raw data or spectra will be sent to scientist/experts at the federal level for analysis and verification. Additional data to be also sent to assist in the verification process include, environmental data, imagery, and background readings. If a detected source has been verified and poses a real threat an emergency situation is assumed and the local Emergency Office and Control (EOC) publishes an alert to the appropriate local, state, and federal tiers based on the severity of the threat via VMART EMS. This is illustrated in steps 1 through 3 of Figure 1. Local, state and federal assets will be deployed in response to the emergency. Emergency protocols and appropriate response actions will be activated at each of the agencies. This is illustrated in steps 4 through 5 of Figure 1.

Another aspect, as part of VMART EMS, involves considering constraints governing resources (web services) that are available to different participating agencies. Figure 2 illustrates part of the motivating scenario as a WSP that avails the participants of VMART EMS with environmental data, imagery, and background readings resources. The availability of the resources is regulated by predefined policy constraints. For example, the local fire department does not have access to onsite video stream. Also the collaborating state transportation agency does not have rights to make use of actual sensor data for further analysis and modeling, whereas the participating federal agency utilizes both of these resources. The interplay of service policies and subsequent services available to the requestor is determined using goal reachability analysis.

Specification of a WSP to represent the above scenario is an interesting challenge. The scenario represents a complex WSP that comprises of several interacting detailed processes involving multiple agencies and different types of resources, e.g., the threat detection and verification by the Threat Detection Service (TDS). The TDS is made up of multiple interacting component services as illustrated in Figure 2. It is a challenge to provide a methodology makes it convenient for a domain expert / system user, who is not
an expert in process specification, to represent the specification of tasks detailing protocol behavior in terms of a set of related scenarios. A scenario corresponds to a sequence of interactions describing how the system reacts to the stimuli from its environment, as illustrated in Figure 3. There is also a need to provide the methods and tools that confirm valid invocation ensuring goal reachability, based on user credentials, of relevant web services that achieve the desired functionality to the tasks of the process.

2. Preliminaries

Soon et al. considers service policies, service flow policies and user policies in [7]. Building on the policies imposed by different entities while composing service flows in [7], we define the policy constraints associated with a WSP as follows: $p = (\{w_1, ..., w_n\}, S)$ where each $w_i$ is a component web service of the process flow $p$ and $S$ is a set of associated policy constraints denoted as $S = \{SP, CP\}$ where $SP$ denotes the associated service policy constraints and $CP$ denotes composition policy constraints. In the following we describe the service policy and composition policy constraints associated with the component web services.

2.1 Service Policy Constraints

The service policy constraints ($sp$) are specific to each individual component web service of the composed process flow. The $sp$ utilizes the credentials of the service requestors to determine the access to specific web service or a set of web services. The credentials of the service requestor are requestor attributes that are essential for access control. These attributes can be as simple as the name of the service requestor, e.g., the name and function of the requesting agency, or can be the role performed by the service requestor, e.g., a collaborating state agency viz., police department, may be delegated a response role in a VMART. A service policy constraint on a component web service may be specified as a set of credentials and/or a set of specific input parameters required for invoking the service. By utilizing credentials, it is possible to enforce critical security policies such as the streaming camera feed is allowed to be viewed only by agencies assuming a security role in a VMART.

The service policy, in [7], is stated in terms of the conditions and contexts under which a particular policy becomes relevant. Our formal definition of the service policy constraints for a composed process flow is as follows: $SP = \{sp_1, ..., sp_n\}$ where $sp = \{cr_{i_1}, ..., cr_{i_l}, [r_1, ..., r_n]\}$ denotes the set of all credentials relevant for the composed process flow. Each service requestor ($sr$) possesses a set of credentials denoted as $sr_{ij}$. A service policy constraint can be expressed as 1) Each component web service $w_i$ of $p$ is associated with a set of credentials, denoted as $(w_i)sp$. 2) A set of component web services $(w_i, w_j, ...)$ is associated with a set of credentials, denoted $(w_i, w_j, ...)$sp. The first set policy constraints is to specify that a service requestor must possess $(w_i)sp$ to invoke $w_i$. Thus, when a component web service is requested by a service requestor $sr$, requestor credentials are checked against the service policy constraints, for $w_i$, to $sr$ only if $(w_i)sp \subseteq sr_{ij}$. The second set of service policy constraints is to specify that the service requestor must possess $(w_i, w_j, ...)$sp to invoke $(w_i, w_j, ...)$ together, e.g., viewing an onsite streaming video along with the relevant sensor readings may require a higher set of credentials.

2.2 Composition Policy Constraints

Service flow policy, in [7], refers to a policy statement to denote a condition to be satisfied to instantiate a service flow. Correspondingly, our composition policy constraints ($cp$) include service type constraints, inter-service constraints and security constraints, specific to each individual component web service of the composed process flow. The $sp$ makes use of the characteristics of the component web service to determine the execution of specific web service or a set of web services. The formal definition of the composition policy constraints for a composed process flow is as follows: $CP = \{cp_1, ..., cp_n\}$ where $cp = \{st_1, ..., st_n, [se_1, ..., se_n]\}$ denotes the set of all execution characteristics relevant for the composed process flow.

Service Type Constraints: The service type constraints specify the requirements on the type of web service, e.g., atomic or complex web service, to be rendered. Formally, we define service type constraints as $(w_i)st$ for $p$. For example, additional weather reporting services may be requested for as an atomic web service.

Inter-Service Execution Constraints: The inter-service execution constraints specify the various temporal relationships that must be adhered to by the component web services when executing the process flow. The various inter-service execution constraints among the component web services may be modeled utilizing interval temporal logic. Atluri et. al. identify possible relationships between...
two distinct intervals including before, later and immediately after, etc [4]. We utilize the model presented by [4] to for capturing the inter-service execution constraints.

3. Our Approach

Current SOA-based application platforms provide independent means of service composition and goal reachability analysis whereas goal reachability analysis of WSPs involves issues related to state, behavior and identity [7, 8]. To extend this paradigm, we utilize HMSCs for specification of the WSP prior to its actual execution with an aim to evaluate their local transitions, states and behavior before actually executing any sub processes thereby evaluating the true request flow. This initial goal reachability analysis of the WSP facilitates the identification of parts of the process flow that may have been implemented incorrectly and may have unforeseen results. As a next step, CP Nets provide an operational level semantics formalism of HMSCs and facilitate the derivation of a deterministic automata model from diagrammatic specification techniques. Finally, the formal specification of processes and interaction protocols is achieved through BPEL [5]. BPEL facilitates the modeling of the behavior of executable processes that model actual behavior of a participant in an interaction and abstract processes. Abstract processes, in contrast, process descriptions for protocols that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior.

Following is an outline of the key steps of our approach, followed by a detailed discussion of each of the steps: 1) HMSC Creation, 2) Mapping components to CP Nets, 3) BPEL Transformation.

3.1 HMSC Creation

A basic MSC is a tuple [13, 16], M = (E, P, g, h, m, <), where E is a finite set of events, P is a set of processes, g is a mapping that associates each event with a process, h is a mapping that associates each event with a type (send or receive), m is a bijective function that maps send events to the receive events. < is a partial order relation between events such that: ∀ e ∈ E: h(e) = send ⇒ e < m(e)

i.e., a send event must be ordered before the corresponding receive event.

In this section we focus on the graphical representation of the HMSC and the corresponding MSCs related to the motivating scenario from the Homeland Security domain. MSC is a member of a large class of specification techniques, e.g., UML Interactions that utilize diagrammatic representation for process specification [13, 12]. The choice for using MSCs over UML in this paper is motivated by the following factors [10]; 1) MSC is a language that considers the graphical syntax tightly coupled to the domain concepts; 2) MSC has more formally defined semantics as compared to UML.

Figure 3 depicts the MSC for a simplified version of the Sensor Controller service described in the motivating scenario. The MSC controller describes the sequence of events where a threat is detected and consequently an alert is issued. The exceeding of a sensor threshold induces sending a message comprising of the sensor data to the sensor data integrator. The sensor data integrator in turn triggers a message for the data to be verified with the help of a mobile sensor. The verified sensor data is then messaged back to the sensor data integrator. The sensor data integrator passes the indication of a threat to the sensor controller. The sensor controller relays the inconclusive spectra for further analysis to the sensor (spectral) analyzer. The spectral analyzer messages back to the controller with the results and also logs the results with a message to the environment. The sensor controller instance is associated with a timer that sends out the threat alert without the specific spectral analysis results. If the spectral results are obtained within the subsequent time out signal of the timer, the controller issues a threat alert combined with the local action of gathering additional information that includes associated environmental data and imagery. The threat alert is then issued to the local agency.

HMSC supports the abstraction of the MSC specifications and services as a guide for linking the modularized MSCs. HMSC specifies the sequences in which the MSCs can be executed. The controller in the HMSC depicted in figure 4 is represented by the MSC presented in Figure 3. Similarly the analyze and alert

![Image](image-url)
3.2 HMSC to CP Nets

In this section we present a formal model, based on Petri Nets to represent the policy constraints defined in section 2. In order to effectively represent the policy, service policy and composition policy, constraints for a WSP, we propose to use an extension of the color-time Petri Nets described in [4].

Petri Nets present an explicit description of both the states and the actions of the system simultaneously. The states of a Petri Nets are represented by places and the actions by transitions. Each place contains a set of markers called tokens. CP Nets extend Petri Nets by allowing the use of tokens that carry data values thereby providing a concise method for modeling and analyzing complex systems. CP Nets combine the strength of Petri Nets that provides the primitives for describing synchronization of concurrent processes with programming languages that provides the primitives for defining data types (color sets) and manipulating data values [11].

Unlike traditional PN Atluri et al. use two types of tokens: a regular token (represented as a filled dot) and a hole (represented as a circle). When a token enters a place, it is said to be available after the time of interval of the place has elapsed. On the other hand, when a hole enters a place, it becomes available as soon as it enters. As far as the dynamic behavior of the Petri Nets is concerned, tokens and holes behave the same way.

A CP Net is a tuple \( N = (\Sigma, P, T, A, N, C, G, I, M, H, D, i, l, e) \) satisfying the following requirements:

- \( \Sigma \) is a finite set of non-empty types called color sets (or simply colors).
- \( P \) is a finite set of places, \( T \) is a finite set of transitions, \( A \) is a set of arcs such that: \( P \cap T = A \cap P = A \cap T = \emptyset \).
- \( N \) is a flow relation defined from \( A \) into \( (P \times T)\), \( C \) is a color function mapping each place \( p \in P \) into a color from \( \Sigma \), \( G \) is a guard function. It is defined from \( T \) into expressions such that:
  \[ \forall t \in T, \text{Type}(G(t)) = B \land \text{Type}(V ar(G(t))) \subseteq \Sigma \]
- \( E \) is an arc expression function. It is defined from \( A \) into expressions such that:
  \[ \forall a \in A, \text{Type}(E(a)) = C(p(a)) \]
- \( i \) is an initialization function mapping each place \( p \in P \) into a multiset over \( C(p) \), \( M = P \rightarrow \Sigma^* \) is the marking, there exists a set \( \{ \text{token, hole} \} \) of types, and a type function \( H \), such that \( \forall p \in m(p), H(p) \in \{ \text{token, hole} \} \).
- \( D \) is a duration function, \( D : P \rightarrow \mathbb{R}^+ \) where \( \mathbb{R}^+ \) represents the set of all real numbers, \( | a \geq 0 |, i, l, e \) are labeling functions.

The above definition states that each marking belongs to either of the two types, token or hole, and is associated with a color (or type) which is defined in the color set \( \Sigma \). Each place is associated with a color set (i.e., denoted as \( C(p) \)), which specifies the set of allowable colors of the tokens (holes) to enter the place.

Given a transition \( t \), we denote its input and output set of places by \( \ast \) and \( \ast \) respectively. A transition \( t \) is enabled only if all of its input places \( p \) are marked, and the tokens (holes) are available. An enabled transition fires. As a result, when more than one input place exists, the transition fires after the maximum duration of all the input places has elapsed. Upon firing, \( t \) consumes all the tokens (holes) from each of its input places \( p \) and deposits them into each of its output places, as their color sets permit.

3.2.1 Mapping of MSC to Color Petri Nets. In this section we present a novel approach, for specification and goal reachability analysis of WSPs, which employs CP Nets as an intermediate model to support analysis and synthesis.

![Figure 5. Petri Net fragment for sequential composition of SR instance in MSC](image)

**Algorithm: mapMSCtoCPNet**

**Input:** Message Sequence Chart (MSC) \( M \) with events \( E \).

**Output:** Colored Petri Net (CP Net) \( N \).

1. \( \text{begin} \)
2. for each \( \text{MSC} : \ m \in M \) do
3. \( \text{map top of MSC to start place} \)
4. \( \text{map bottom of MSC to end place} \)
5. \( \text{/* map e from transitions to MSC events */} \)
6. \( \text{map places and transitions to instance names} \)
7. \( \text{//map controlling labels to message identifiers} \)
8. \( \text{//map sending and receive events */} \)
9. \( \text{create additional place to start place for receive event, E} \)
10. \( \text{create additional place to end place for send event, E} \)
11. \( \text{Fuse start place of E with end place of subsequent E} \)
12. \( \text{sequentially append instances transitions in the MSC} \)
13. \( \text{end for} \)
14. \( \text{return N} \)
15. \( \text{end} \)

![Figure 6. mapMSCtoCPNet algorithm](image)

mapMSCtoCPNet algorithm shown in Figure 6.

Figure 7 depicts the corresponding CP Nets for the Homeland Security scenario MSC. We make use of service discovery to retrieve the services in the post MSC phase. Appropriate services are discovered to fulfill the functionality described in the scenario.
This service will be invoked only if a token with color *federal* enters this place. On the other hand, if a hole with color *federal* enters, the service will not be invoked.

Mapping of WSP to include the policy constraints makes use of the CP Nets derived in the previous section. Next we identify three distinct color sets: $\Gamma$, $\Lambda$, and $\Omega$ such that $\Gamma \cup \Lambda \cup \Omega = \Sigma$ and $\Gamma \cap \Lambda \cap \Omega = \emptyset$, where $\Gamma$ denotes interservice execution characteristics, $\Lambda$ the different types of services and $\Omega$ the different types of credentials. For example, to determine the color set $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$, we assign a distinct color for each interservice execution characteristic. Similarly, we select the color sets for $\Lambda = \{\lambda_1, \lambda_2, \ldots\}$, and assign a distinct color for each credentials type. This procedure is repeated for $\Omega = \{\omega_1, \omega_2, \omega_3, \ldots\}$, to assign a distinct color for each credentials. Each component web service $w_i$ in the WSP is represented by a place $p_i$, where the color set of $p_i = \{\gamma_i, \gamma_i, \lambda_i, \omega_i, \omega_i, \omega_i, \ldots\}$. The interservice execution characteristics required are represented as tokens of the assigned color in $\Gamma$, the service type is represented as tokens of the assigned color in $\Lambda$, and the security credentials of the service requestor intending to invoke the service as tokens of the assigned colors in $\Omega$. The tokens and holes into $p_{initial}$ is such that the total number is equal to the number of distinct colors used. A guard function, $G$, attached to a transition node is to provide a binding, for firing in the enabled transition node set, for the same user. If the guard function of the enabled transition is evaluated to be true to taking action, the transition is activated. The action of the activated transition can be fired and pass the tokens from its entering places to its outgoing places. For example, only the transition node will be fired where in user $u_x = u_y = u_z$. The request is checked by the guard, which evaluates to true if it has tokens corresponding to the same users, e.g., $<u_x, bk> <u_y, gr>$ represent the credentials security and federal associated with the same user, as illustrated in Figure 8. The guard function also enforces that users with inadequate tokens are not able to invoke the service.

Figure 8 illustrates the policy constraints for motivating scenario that utilizes web services as explained in section 2. Each place, in figure, is represented with a set representing the service policy constraints. For example, $w(p_3) = \langle rd, bk, or, gr \rangle$ from the credential set of $\{bk, cy, gr, pi\}$, as illustrated in Figure 8. This dictates that the component web service can be invoked by an agency in a security role $\{bk\}$ and at the federal level $\{gr\}$. $\langle rd \rangle$ represents an atomic web service, characterizing its interservice execution constraint. The credentials and characteristics required are represented as tokens and absence of them is represented as holes. The initial places are marked with these to start with. For example in the figure shown, the service requestor is not performing a security role at the federal level and therefore the initial marking contains holes for colors $\{bk\}$ and $\{gr\}$. Also within a constraint type, one constraint property can inherit the properties of others, such as the credential $\{yl\}$ (Fire...
Department) inheriting the property of the corresponding \( \{cy\} \) (VMART). The higher level credential is then represented with a color set that includes all the lower level credentials. For example for the fire fighter responder credential is represented with two tokens \( \{yl, cy\} \). The CP Nets representing the WSP is designed in such a way that, when it is inserted with the tokens and holes that reflect the credentials and characteristics into the initial place, it reflects the specific goal reachability of the composed WSP. For example if a service requestor’s credentials are and characteristics are health department, member of VMART as a local agency, requiring atomic service, time duration 25, and no access to critical security components. With this marking holes will be entered in places \( p_4 \) and \( p_5 \), thus making these component web services inactive, and therefore, these will not be invoked.

With this marking holes will be entered in places \( p_4 \) and \( p_5 \), thus making these component web services inactive, and therefore, these will not be invoked.

**Figure 8. Policy Constraints for Threat Detection Service for the Homeland Security motivating scenario.**

The goal reachability analysis of the process, thus, involves the creation of initial markings for a process run to check for consistency and accuracy of execution. In our work we focus on problems that include reachability analysis, and checking for liveness and existence of deadlocks, assuming a finite domain. The reachability problem pertains to deciding if a marking M of N is reachable for a given net N. Reachability analysis for a given process ensures that the process does not reach an unsafe state \[11\]. An unsafe state pertains to all states that do not yield the minimum output as desired by the service requestor. The specification and analysis of the minimum output in terms of the specific services invoked and/or the output parameters desired forms part of our future work.

### 3.3 BPEL Transformation

BPEL extends the web services interaction model and supports inter-organizational processes by providing a formal specification of the interacting processes and the interaction protocols. BPEL is a Turing complete language comprising of expressive XML representation of advanced constructs that supports graph-based and structured modeling \[5\]. Our work utilizes the approach provided by \[1\] to transform the theoretical foundation of CP Nets as a source language to the block structured BPEL as a target language. Transforming a graph structure to a block structure involves the mapping of graph-based CP Nets onto a hierarchical decomposition of specific BPEL construct utilizing CP Nets fragments with a well defined start and end, e.g., a sequential fragment maps to the sequence construct where as a parallel fragment maps to a flow construct. Within the fragments, transitions are marked with references to BPEL primitives, e.g., invoke - an operation call to a web service, receive - receive a message from a resource, reply - send a message to a resource, wait - induce a time delay, assign - transfer data, throw – identify exceptions, and empty - null operator. The CP Nets instances are then mapped onto BPEL structures making use of these constructs. Our current work achieves a semi-automated transformation by providing a procedure to map CP Nets to BPEL code structures as illustrated in Figure 9. The actual complete BPEL code, however, has to be completed by using these structures as a guide.

**Figure 9. BPEL structure for SR instance.**

In this paper we propose and investigate an approach that combines effective modeling, appropriate formal framework and an implementation process for dynamic WSP and complex web service composition and goal reachability analysis. Most of the existing approaches do not provide a comprehensive unified methodology for an appropriate design representation and formal specification. In \[19\] Yang et al present an approach to verify web services composition and utilize transformation rules to transform service composition proposals into CP Nets for further performance analysis. Yang et al utilize an UML Activity Diagram as a model to illustrate web service composition. Their approach, however, does not provide a mapping from the composition model to the formal CP Nets model and also does not generate the corresponding BPEL for the verified model. Yi et al. \[20\] propose methods to integrate a partner service with a complex conversation protocol into a composition of web services following a formal verification of the correctness of the composition. They, however, do not consider complex WSPs and consequently do not provide a diagrammatic representation and its initial verification. The approach presented in \[1\] is more analogous to our work. This work describes a case study where for a new bank system requirements are mapped onto Colored Workflow Nets which are then implemented using BPEL in the IBM WebSphere.
environment. One of the listed drawbacks of EUC, which is the basis for the requirements model, is that it potentially requires an extra effort during specification of user requirements as a formal language is not ideal for communication with users. In contrast our approach begins with an efficient diagrammatic representation by system users of the WSPs and moves on to a more formal version where precision and goal reachability analysis is added. In [14] Narayanan et al. describe the web service composition in terms of input output composition of an atomic service in a Petri Nets formalism. Our work presents the next step considering the general process composition where in we consider all possible interleaving of the component services in a WSP. Our work describes the instantiation of the general process that is specific to individual users. The process instantiation takes in to account the dynamicity aspects in terms of the changes in the domain requirements, the user request and the component services that form the WSP.

5. Conclusion and Future Work

In this paper we presented an approach that combines effective diagrammatic modeling, appropriate formal framework and an implementation process for dynamic WSP and complex web service orchestration and goal reachability analysis in the Homeland Security domain. Specifically we introduce a new approach for WSP specification and goal reachability analysis comprising of service orchestration using HMScs and CP Nets that provides a methodology for specification at a process level as well as goal reachability analysis at the service level. We also present an approach to create BPEL processes from the verified CP Nets. We believe that this approach is generalizable and can be applied to other domains that involve complex interactions.

As future work we plan to investigate annotation techniques for CP Nets. We also plan to extend the BPEL mapping to provide specific structures for occurring CP Nets patterns. Extension of our approach to synthesize CP Nets mapped to broader library of HMSc patterns also forms part of our future work.

6. References


