A CPN Provenance Model of Workflow: Towards Diagnosis in the Cloud

Yingmin Li and Omar Boucelma

Laboratoire des Sciences de l'Information et des Systèmes,
Domaine Universitaire de Saint-Jérôme
13397 CEDEX 20, Marseille, France
{yingmin.li,omar.boucelma}@lsis.org

Abstract. Workflow provenance is an important supportive component that encompasses knowledge sharing, product reusability and process verification. The emerging cloud computing paradigm offers new application opportunities but also raises research challenges, such as integrity, privacy, security and legal related issues. In this paper, we propose a Colored Petri Net (CPN) model for diagnosis based on Open Provenance Model (OPM). An illustrative application is presented: a workflow is expressed as a composition of services deployed in the Cloud, and security is implemented by means of Web Service Security policies (WS-S).

Keywords: provenance, OPM, workflow, cloud computing, WS-S, CPN, diagnosis

1 Introduction

The provenance of workflow focuses on recording the data manipulations that happen along the computational steps to answer the questions like: Who created this data product and when? When was it modified and by whom? What was the process used to create the data product? Were two data products derived from the same raw data [1]? These questions arise the issues concern data integrity, privacy, security and access control. The provenance data can help to ensure the data re-productivity, data quality determination, fault-prone detection(verification) and diagnosis [4, 5] with the support of a formal model and communication protocols. Due to the loosely-coupling nature of workflow components, they are usually modeled as Discrete Event Systems (DES) like Directed Acyclic Graph (DAG), Open Provenance Model (OPM) or Petri net.

Cloud computing, as a potential momentum of Internet infrastructure revolution, brings to light the springing up of workflows. When the components runs on the un-trusted clouds, some existing provenance issues, like privacy, security, access control, become unavoidable and new related provenance challenges are arisen, such as provenance availability, extensibility [2] and scalability in the cloud environment [3, 4].

In this paper, we model OPM as Colored Petri Net (CPN) for the purpose of Model-Based Diagnosis (MBD) in the cloud. We introduce the security properties [6–8], which is not included in OPM but mandatory for the cloud-based provenance. More specifically, we model the Web Service Security (WS-S) protocols as a part of OPM model for CPN diagnosis. A novel diagnosis approach is proposed based on the CPN diagnosis model of [9].

The remainder of the paper is organized as follows: Section 2 details a motivating example. Section 2 introduces the OPM model and shows how to incorporate WS-S protocols into OPM. Section 3 describes the CPN formal background together with OPM to CPN translation, while Section 4 sketches the diagnosis approach and algorithm. Finally we conclude in Section 5.

1.1 Motivation example

Suppose a very simple workflow as illustrated in Figure 2, a paper search engine workflow, which has three functions (emphasized by black rectangles), and locates in two different clouds (framed in the dotted rectangles C_1 and C_2). In cloud C_1, operation *search* takes the title of the paper (*title*) as input, if the search engine finds the related papers, it returns the paper details such as bibtex (*bib*), PDF address ($PDFAdd$) and other abstract information (*Abs*), otherwise, it returns null result. If the user asks for further information of $PDFAdd$ or *Abs*, the workflow continues to run in C_2. The data transported between cloud C_1 and C_2 uses WS-S protocol, so it requires the username/password (*usr*/*psw*) to encrypt $PDFAdd$ or *Abs* on C_1, transport the encrypted $S-ADD$ or $S-Abs$ and then to decrypt to get $PDFAdd$ or *Abs* on C_2. In C_2, the username/password (*usr*/*psw*) and the IP address is required for authorization and returns the PDF file (PDF) and the references (Ref) of the paper. In a scenario, a user input a *title* and then *usr*, *psw*, but did not receive PDF while got the correct Ref. In this case, s/he required to precess diagnosis for the workflow.

2 Workflow provenance with end-to-end security

The end-to-end security is an essential pre-condition for offloading the data storage and handling to the cloud providers. Concerns the workflow security, integrity violence of data and malicious software components are the most common threats, which provence should serve to prevent or trace. A straight way to run the workflows in the cloud is to use Service-Oriented Architecture (SOA) and ensure the secure communications through appropriate protocols. WS-S is a flexible and feature-rich extension to Simple Object Access Protocol (SOAP). It describes how integrity and confidentiality can be enforced on messages. In this section, we instantiate the end-to-end secure provenance-like issues proposed in [10] with OPM and WS-S protocols.

2.1 OPM

OPM is a multi-layer provenance model which describes the data and activities dependencies of all the participants. In essence, it consists of a directed graph expressing such dependencies. OPM [11] defines a set of nodes: *artifact* (an immutable piece of state), *process* (an or a series of activities) or *agent* (contextual entity acting as a catalyst of a process) and a set of dependencies between them such as (*process*) *used* (*agent*), (*artifact*) *wasGeneratedBy* (*process*), (*process*) *wasControlledBy* (*agent*), etc. These causal dependencies are represented as the edges between the entities from the effect to the cause. The dependencies are composable: e.g., (A) wasGeneratedBy (P1) + (P2) used (A) = (P2) wasTrigeredBy (P1). OPM defines *roles* of agents or artifacts as the constituents of the dependencies mentioned above to distinguish them when multiple dependencies edges are connected to different processes. OPM defines *accounts* as the leverage of the dependencies: the data dependencies can *co-existing* and be *overlapped.* A finer-granulated dependency is a *refinement* of the less finer-granulated one. A *profile* specifies the semantics of OPM, which defines the controlled vocabulary, patterns and inference rules. OPM introduces the optional observation times (*creation* and *use* for *artifacts* and *starting* and *ending* for *process*) of its entities. These temporal constraints should be compatible with the causal dependencies.

OPM defines *annotation* as an *extra information* for inter-operability purpose, which allows meaningful exchange of provenance information. OPM allows all its entities to be annotated as multiple property-value pairs which are associated with multiple instances and can be self-annotated. OPM also defines *profile* and a series of compatible *profile expansion* rules for the communities to develop their own concrete practice and usage guidelines. But the profiles are assumed to be specified in natural languages and expanded manually.

The difference between OPM and other DES is that OPM models the dependencies backward. The edges are linked from effect to cause which is intuitive for workflow provenance.

2.2 WS-S protocol

For the data oriented workflows, which runs in the cloud computing environment, the security problems are always the top concerns. So security data transformation is necessary, so their provenance should take security problem into consideration. We choose WS-S [12] as a representation of the secure data transformation protocols because it supports lots of provenance requirements and this choice doesn't lose the generality of OPM model. Our idea is to model the WS-S protocol as OPM in a general way so as to integrate the indispensable secure communications in the cloud computing environment.

A typical WS-S protocol example is as follows: the client first sends a request to ask for a security token; the security token service returns a token (e.g., username/password) to the client; the client signs and sends the token to the Web service; the Web service validates the token with the security token service; once the token is validated, the Web service responds to the client.

Figure 1 is the OPM of this WS-S protocol. Two agents, security token service ($cert - Center$) and Web service (WS) use two processes: *distribute* and *authorize* to respectively distribute the token and authorize the client participate the provenance. Process $WS - function$ is the real WS operation. The

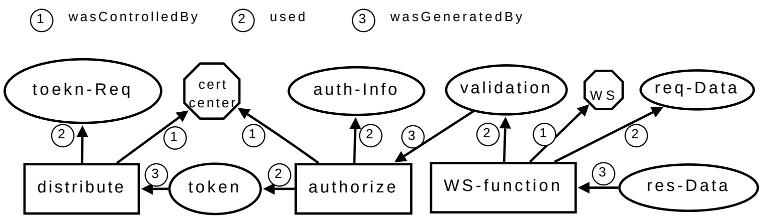

Fig. 1: A representative message flow of WS-S protocols

dependencies between the entities and the related roles are annotated on the edges. There are several similar WS-S message flows [13] which can be easily translated into OPM, so we omit it here.

2.3 Example: OPM

The OPM model of the given example is illustrated in figure 2. There are three agents: engine (*Eng*), client (*Cg*) and authorization agent (*Ag*). On each edge, the dependencies and the related roles of the agents are annotated (e.g. *search* and *getPDF* processes are *controlledBy* agent *Eng*).

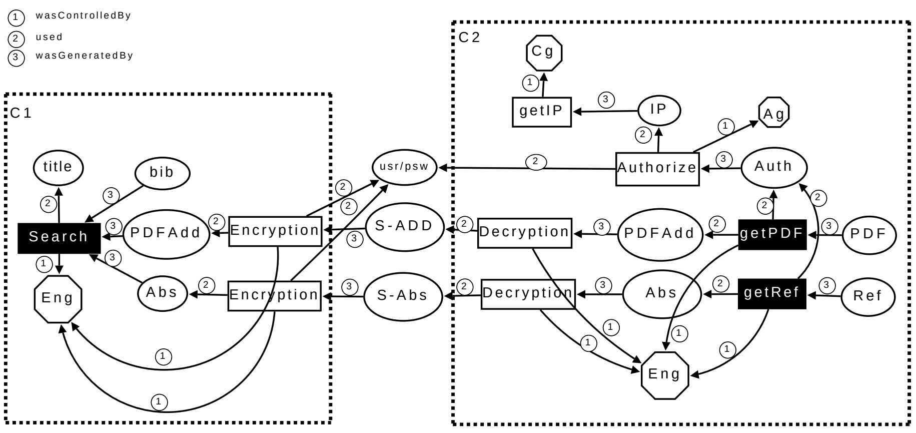

Fig. 2: An OPM illustration of the paper searching example

3 CPN fault model

Colored Petri net is a bi-part graph model of DES which has the rich data expressive capabilities. Basically, it contains *color sets* (and colored functions over them), which represent system properties; *places*, which can contain colored tokens to represent the system status; *transitions*, which represent the system events; *arcs*, which connect the places and transitions to represent the causal relationships.

Many formal analyzing approaches are created for design, specification, simulation and verification of systems. CPN can simulate the workflow executions or its provenance by *firing* from one marking (system state, see definition 2) along the (concurrent) execution paths. The hierarchical definition of CPN *pages*, which relate a transition (and its surrounding arcs and places) to a lower level CPN. The low level CPN provides a more precise and detailed description of the activity. This hierarchical structure is suitable to model the fine-granulated dependencies for provenance management and to scale up in the cloud. So we choose CPN as the formal analysis model of OPM for diagnosis.

In MBD literature, diagnosis is to construct an abstract model of the system to describe the (correct and/or faulty) behaviors of the system and compare them with the observations to deduct the fault from the deviations. The fault(s) causes are of various sources: 1) the flawed initial data, 2) dysfunctional process or 3) authorization problems. We can abstract these possibilities and construct a simple fault model (see definition 1): to distinguish the correctness status of data, process and user. The correctness of data or authorization information of user are represented as the correct (b) and faulty (r) color of CPN places; the correctness of process is represented, in the CPN model, by the color of an additional place, of which the color is decided by the color(s) of the input place(s). The correctness unknown status of places is represented by *unknown* color ($*$).

Definition 1 (CPN fault model graph) *A CPN model [9] is a tuple* $N = \langle \Sigma, \Gamma, P, T, Pre, Post \rangle$*, where:*

- *(i)* Σ *is the set of colors types* $b, r, *$*;*
- *(ii)* P *is a set of labeled places of type* Σ*:* $\forall p \in P$, $cd(p) \in \Sigma$*;*
- *(iii)* T *is a set of labeled transitions;*
- *(iv)* $cd : P \to \Sigma$ *is a color domain for places;*
- *(v)* $Pre \in \mathbb{B}^{|P| \times |T|}$*: is the forward matrix consists of color variables and color functions, where* $\mathbb{B} = \bigcup_{\sigma \in \Sigma} [\sigma \to \Psi\langle\sigma\rangle]^1$*;*
- *(vi)* $Post \in \mathbb{B}^{|P| \times |T|}$*: is the backward matrix.*

Definition 2 (CPN marking) *A* marking M *of a CPN graph is a multi-set vector indexed by* P*, where* $\forall p \in P$, $M(p) \in \mathcal{M}^+(cd(p))$*.*

Definition 3 (CPN system) *A* Colored Petri Net System *(CPN-S) is a pair* $S = \langle N, M \rangle$ *where* N *is a CPN graph and* M *is one of its marking.*

A CPN-S is a dynamic system and each dynamic state is represented by the marking of the system. When a CPN-S is *enabled* (see definition 4), it is ready to *fire* (see definition 5).

[1] we use the notation $[D \to D']$ to represent the set of all application from D to D'.

Definition 4 (CPN enabling rules) *Let S=$\langle N, M\rangle$ be a CPN system, t be a transition in N, a transition t is enabled, noted $M[t\rangle$, iff $\exists\beta$, with $M \geq Pre[.,t]^{\beta}$.*

Definition 5 (CPN firing rule) *Let S=$\langle N, M\rangle$ be a CPN-S, t a transition, with $M[t\rangle$ for some β. The firing of the transition t changes S to $S' = \langle N, M'\rangle$ with $M' = M + C(.,t)(t)^{\beta}$. We denote the firing as $M[t\rangle^{\beta}M'$.*

3.1 Translation from OPM to CPN

Intuitively OPM has much similarities with CPN, and in the sense of data oriented-workflow provenance, CPN keeps the same expressive capability concerns the structural, and data/control dependencies representation. Most OPM concepts can be formally mapped to CPN definitions even though OPM is less formally specified then CPN definition. The mapping from OPM to CPN is as follows: to translate

1. an OPM artifact or agent to a CPN place, an OPM process to a CPN page, an OPM edge to a CPN arc, an OPM role to a CPN token color;
2. add a CPN color variable χ_a on the arc $a \rightarrow p$ to represent the color of OPN artifact a when it was *used* by p;
3. the OPM dependency $wasDerivedFrom$ to a composable CPN dependency function $wasDerivedFrom(\chi_{in})$ on the arc $p \rightarrow t$ (or $wasGeneratedBy$ edge in OPM) to represent the dependency between $\chi_{out'}$ and χ_{in}[2];
4. the OPM dependency $wasControlledBy$ to a CPN dependency function $wasControlledBy$ which is a constant function;
5. an OPM account to a CPN firing instance.

For the moment, we define the agent is not responsible for fault, so the output of function $wasControlledBy$ is always $*$ (see table 1). The color $*$ is compatible with b or r, but b and r are not compatible with each other, which reduces the set of fault(s). In OPM, one artifact can be *used* by different processes (e.g., the security token usr/psw is used by both $encryption$ and $authorize$ processes). If these processes are executed concurrently, the correctness status of the artifact is not obvious. In this situation, we add, in CPN model, the additional places (e.g., usr/psw_1, usr/psw_2) to record the temporal status of the place. We also add a $reassure$ transition to take these additional places as input and the original place (usr/psw) as the output. We define a data dependency function $wasReassuredBy$ to represent the color of the input places are reassured by the added transition (see table 1, e.g., $wasReassuredBy(b, r) = b$).

By firing the CPN fault model, we can deduct the possible origins of fault(s) from the symptoms. The edges in OPM is from the result to the cause, as well the corresponding arc in its CPN model. So diagnosis can be realized by firing the CPN from the symptom marking which represents the symptom of the workflow (detailed in section 4). Then the final marking can represent the diagnosis result.

[2] When there are multiple input places, the corresponding color variables of all each input place are the parameters of $wasDerivedFrom$ function, so faults spread through the color variable χ and the dependency function $wasDerivedFrom$

χ_{in_1}	χ_{in_2}	χ_{out}		
		wasDerivedFrom	wasReassuredBy	wasControlledBy
b	b	b	b	$*$
r	r	r	r	$*$
$*$	$*$	$*$	$*$	$*$
b	r	r	b	$*$
b	$*$	$*$	b	$*$
r	$*$	r	r	$*$

Table 1: Value tables of data dependency functions

3.2 Example: CPN fault model

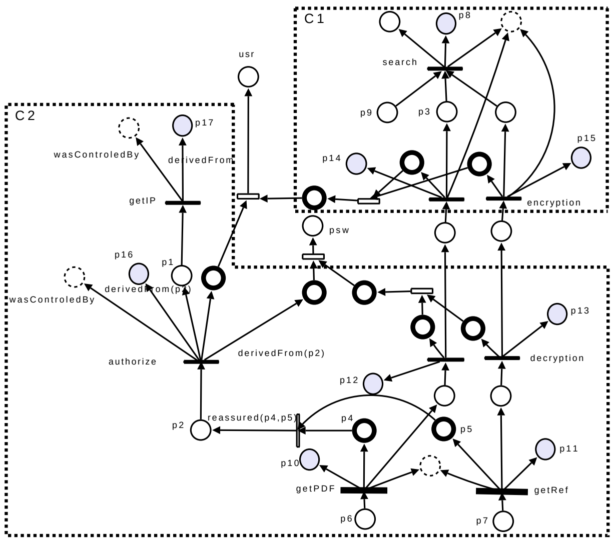

Fig. 3: A CPN illustration of the paper searching example

Figure 3 illustrates the CPN fault model of the paper searching example. The dotted places represent the agents, the solid places represent the places of correctness of transitions and the hollow places represent the artifacts; the solid transitions represent the process and the hollow transitions represents the "*eassure* transitions. Most arc expressions are omitted for visibility. Place p_1 represents the IP address, p_2: authorization validation, p_3: PDF address, p_4: duplication of p_2 which corresponds to $getPDF$ transition, p_5: another duplication of p_2 which corresponds to $getPDF$ transition, p_6: PDF file, and p_7: references, $p_8 - p_{17}$: correctness of transitions. Several $reassure$ transitions (represented as hollow rectangles) reassure the diagnosis, e.g., the transition between p_2 and p_4, p_5.

4 CPN Diagnosis

The mathematical properties of CPN are summarized in initial marking M_0, final marking M_n, characteristic vector $\overrightarrow{\delta}$ (see definition 7) and incidence matrix C (see definition 6).

Definition 6 (Incidence Matrix of CPN) *To each CPN graph, associate its terms incidence Matrix* $C = Post - Pre$.

Definition 7 (CPN characteristic vector) *Let* $\delta \in T^*$ *be a sequence of observed transitions of a net, its* characteristic vector $\overrightarrow{\delta} : T \rightarrow \mathbb{N}$ *which represent the occurrence of each* t *in* T^{*}[3].

Given the incidence matrix of CPN, an initial marking and a characteristic vector, the corresponding final marking is calculated by the incidence equation (see equation 1). In the CPN fault model, the initial marking is a symptom marking, the characteristic vector is an observation occurrence vector indexed by the transition, and in the final marking, each red token represents one fault symptom, so the symptom marking can represent multiple faults.

$$\mathrm{M}_n = M_0 + C \times \overrightarrow{\delta} \quad (1)$$

4.1 Diagnosis of multiple faults symptom

Algorithm 1 CPN diagnosis of multiple faults symptom

```
1: for all $i \in N$ do
2:     construct a symptom marking $M_0^i$ with $M_0^i(p_i) = r$, $M_0^i(p_j) = *, (p_j \neq i)$ and $M_0(p_j) = r$; [4]
3:     $Diag_i = \{p | M_n^i(p) = r, M_n^i = M_0^i + C \times T\}$
4: end for
5: $Diag = \overset{\cup}{\times}\, Diag_i$ with $i \in N$;
```

In case a symptom marking contains multiple red tokens, it is necessary to calculate separately. Because by firing the CPN system, the diagnosis result elements have *or* relations between each other. While for the diagnosis results, which correspond to multiple red tokens, we need to combine them with relation *and* with a Cartesian operator (see definition 8). So the diagnosis is calculated in $N + 1$ steps, where N is the number of red tokens in the initial symptom marking (see algorithm 1). The first N steps separately calculate the diagnosis for each single red token symptom, and the $N + 1^{th}$ step combine the results.

Definition 8 (Multi fault operator) $\overset{\cup}{\times}$ *is an operator that calculates the Cartesian product and then keeps the minimal subsets.*

[3] $\rightharpoonup$ represents a vector, which is a column of elements of the same type domain

4.2 Example: diagnosis

According to the description in section 1.1, p_6 is faulty while p_7 is correct, so M_0 contains a r in p_6 and a b in p_7. $\overrightarrow{\delta}$ is a vector in which each transition is fired once. The incidence matrix C records the color variables and data dependency functions (see table 2). By applying the incidence equation 1, the diagnosis result $Diag = \{\{p_8\}, \{p_{10}, \{p_{12}\}, \{p_{14}\}$, which represents the fault can come from *search*, *getPDF*, *decryption* or *encryption*, while the possibilities of fault(s) on *authorize* is excluded by the *reassure* functions.

	getIP	Authorize	search	reassure	getPDF	getRef
p_1	$-\chi_{p_1}$	$wasDerived$ $From(\chi_{p_2})$				
p_2		$-\chi_{p_2}$		$wasReassured$ $By(\chi_{p_4},\chi_{p_5})$		
p_3			$-\chi_{p_3}$		$wasDerivedFrom(\chi_{p_6})$	
p_4				$-\chi_{p_4}$	$wasDerivedFrom(\chi_{p_6})$	
p_5				$-\chi_{p_5}$		$wasDerivedFrom(\chi_{p_7})$
p_6					$-\chi_{p_6}$	
p_7						$-\chi_{p_7}$

Table 2: Incidence matrix

Consider a multiple faults symptom example: both p_6 and p_7 contains red tokens, the diagnosis has three steps:

1. construct a symptom marking $M_0^1(p_6, p_7, p_9) = (r, *, *)$ and calculate the diagnosis result $Diag_1 = \{\{p_8\}, \{p_{10}\}, \{p_{12}\}, \{p_{14}\}, \{p_{16}\}, \{p_{17}\}, \{usr\}, \{psw\}\}$, which represents the fault can come from *search*, *getPDF*, *decryption*, *encryption*, *authorize*, *getIP*, *usr* or *psw*;
2. construct a symptom marking $M_0^2(p_6, p_7, p_9) = (*, r, *)$ and calculate the diagnosis result $Diag_2 = \{\{p_8\}, \{p_{11}\}, \{p_{13}\}, \{p_{15}\}, \{p_{16}\}, \{p_{17}\}, \{usr\}, \{psw\}\}$;
3. combine the results $Diag = Diag_1 \overset{\cup}{\times} Diag_2 = \{\{p_8\}, \{p_{16}\}, \{p_{17}\}, \{usr\}, \{psw\}, \{p_{12}$ (or $p_{13})\}, \{p_{14}$(or $p_{15})\}, \{p_{10}, p_{11}\}\}$ (p_{12} and p_{13}, p_{14} and p_{15} are combined for expressive convenience).

CPN Tools is a tool for editing, simulating, and analyzing Colored Petri nets [14], which can perform diagnosis with the CPN fault model proposed in this paper. The correctness status is defined as enumerate type *COLOR* (b, r, un) and all the data dependency functions are defined based on *COLOR* set.

5 Conclusion

In this paper, we presented a model based diagnosis approach to monitor workflow provenance in the Cloud. A workflow is first specified in OPM, then translated into CPN, hence amenable to diagnosis. Our model also captures the secure communication protocols between the components in order to realize the end-to-end security, instead of point-to-point one, which is suitable for cloud architecture. Because of some blurs of

OPM *profile* definition, it is not clearly included in the CPN fault model, but it can be considered to clarify the diagnosis result, which is still an open question. The next step is to clarify the hierarchial diagnosis architecture and to study how to diagnosis faults that may be caused by malicious attacks by a sibling instance of a same workflow.

References

1. S. B. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1345–1350.
2. R. Spillane, R. Sears, C. Yalamanchili, S. Gaikwad, M. Chinni, and E. Zadok, "Story book: an efficient extensible provenance framework," in *First workshop on Theory and practice of provenance.* Berkeley, CA, USA: USENIX Association, 2009, pp. 11:1–11:10.
3. H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security and Privacy*, vol. 8, pp. 24–31, 2010.
4. K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Provenance for the cloud," in *Proceedings of the 8th USENIX conference on File and storage technologies*, ser. FAST'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 15–14.
5. M. A. Sakka, B. Defude, and J. Tellez, "Document provenance in the cloud: constraints and challenges," in *Proceedings of the 16th EUNICE/IFIP WG 6.6 conference on Networked services and applications: engineering, control and management*, ser. EUNICE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 107–117.
6. R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: preventing history forgery with secure provenance," in *Proccedings of the 7th conference on File and storage technologies.* Berkeley, CA, USA: USENIX Association, 2009, pp. 1–14.
7. U. Braun, A. Shinnar, and M. Seltzer, "Securing provenance," in *The 3rd USENIX Workshop on Hot Topics in Security*, ser. USENIX HotSec. Berkeley, CA, USA: USENIX Association, July 2008, pp. 1–5.
8. B. J. Corcoran, N. Swamy, and M. Hicks, "Combining provenance and security policies in a web-based document management system," in *On-line Proceedings of the Workshop on Principles of Provenance (PrOPr)*, Nov. 2007.
9. Y. Li, "Diagnosis of large software systems based on colored petri nets," Ph.D. dissertation, 2010.
10. J. Bacon, D. Evans, D. M. Eyers, M. Migliavacca, P. R. Pietzuch, and B. Shand, "Enforcing end-to-end application security in the cloud - (big ideas paper)," in *Middleware*, ser. Lecture Notes in Computer Science, I. Gupta and C. Mascolo, Eds., vol. 6452. Springer, 2010, pp. 293–312.
11. L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche, "The open provenance model core specification (v1.1)," *Future Generation Computer Systems*, July 2010.
12. K. Lawrence, C. Kaler, A. Nadalin, R. Monzillo, and P. Hallam-Baker, "Web services security: Soap message security 1.1," *OASIS*, February 2006.
13. M. C. International Business Machines Corporation, "Ws-security appnotes," 2003.
14. M. Westergaard and H. E. Verbeek, "CPN tools," 2010.