Adaptable Situation-aware Secure Service-based (AS^3) Systems

[1]S. S. Yau, [1]H. Davulcu, [2]S. Mukhopadhyay, [1]D. Huang, [1]Y. Yao

[1]Arizona State University, Tempe, AZ

[2]West Virginia University, Morgantown, WV

Abstract

Service-Oriented Architecture (SOA) has the major advantage of enabling rapid composition of distributed applications, and has become a popular architecture of many large-scale service-based systems. In these systems, various capabilities are provided by different organizations as services and interconnected by various types of networks, including wireless (infrastructure or ad hoc) and wired networks. The services can be integrated following a specific workflow, which is a series of cooperating and coordinated activities designed to carry out a well-defined process to achieve a mission goal for users. For large-scale service-based systems involving multiple organizations, high confidence and adaptability are of prime concern in order to ensure that users can use these systems anywhere, any time using various devices, knowing that their confidentiality and privacy are well protected and the systems will adapt to satisfy their needs in various situations. Hence, these systems must be adaptable, situation-aware and secure. In this paper, we outline our approach to enabling rapid development of and provide runtime support for hierarchical situation-awareness capability and policy-based security in AS^3 systems, and enabling users to rapidly generate, discover, contract with and compose reliable and unreliable services into processes to achieve their goals based on the situation and adapt these processes when situation changes.

1. Introduction

Service-based systems are based on Service-Oriented Architecture (SOA) [WSA04], which is becoming more popular for large-scale distributed systems, such as Grid and Global Information Grid (GIG) [D8100.1], in various application domains, including collaborative scientific and engineering work, e-business, healthcare, military applications, and homeland security. A major advantage of SOA is its capability of enabling rapid composition of distributed applications, regardless of the programming languages and platforms used in developing and running different components of the applications. In these service-based systems, various capabilities, including computation, storage, computation and communication, are provided by different organizations as *services*, which are software/hardware entities with well-defined interfaces to provide certain capability over wired or wireless networks. The services can be integrated following a specific *workflow*, which is a series of cooperating and coordinated activities designed to carry out a well-defined process to achieve a mission goal for users. For these large-scale service-based systems, *high confidence* and *adaptability* are of prime concern. It is very important to ensure that users can use these systems anywhere, any time using various devices (ranging from handheld devices, such as smart phone and PDA, to PCs and supercomputers), knowing that their confidentiality and privacy are well protected and the systems will adapt to their needs in different situations. Therefore, these systems must have the following properties:

(1) *Adaptability*. In these systems, services may become unavailable due to distributed denial-of-service attacks or system failures, and new processes may be created in runtime to fulfill users' new mission goals. Hence, the systems must have the capabilities to change their configurations to provide continuous availability, or to adapt their behavior to satisfy the new ~~processes~~ goals in dynamic environments.

(2) *Situation-awareness* (SAW). SAW is the capability of being aware of situations and adapting the system's behavior based on situation changes [Yau02ab]. A *situation* is a set of context attributes over a period of time that is relevant to future system behavior [Yau02ab]. A *context* is any

instantaneous, detectable, and relevant property of the environment, the system, or users, such as time, location, light intensity, wind velocity, temperature, noise level, available bandwidth, invocation of action, and a user's schedule [Yau02ab]. SAW is essential for high confidence and adaptable service-based systems because it is needed for determining adaptive processes to achieve users' goal, and enforcing flexible security policies.

(3) *Security*. To provide high confidence to users, these systems must have the capabilities of authenticating users and service providers, verifying the integrity of services, protecting the confidentiality of information, controlling the access to services based on security policies, and detecting malicious services and users.

Although various techniques have been proposed to improve the security and dynamic service composition of service-based systems [BPEL4WS, WSS, WSSP, XACML], so far there are no effective enabling techniques for *Adaptable Situation-aware Secure Service-based* (AS^3) *Systems*. In this paper, we will discuss the requirements of AS3 systems and outline our approach to achieving the following objectives: (1) Enable rapid development of and provide runtime support for *hierarchical situation-awareness* and *policy-based security* in AS^3 systems. (2) Enable users to rapidly generate, discover, contract with and compose reliable and unreliable services into processes to achieve their goals based on the situation and adapt these processes when situation changes. An example is given illustrate the requirements of AS^3 systems and the advantages of using our approach to addressing these requirements.

2. Requirements of AS^3 Systems

Before presenting the requirements of AS^3 systems, let us consider a global service-based system which system connects various units, such as aircrafts, ships, offices, etc, through wired and wireless networks. Each unit may provide certain services to allow other units to utilize some of its capabilities. ***Figure 1*** illustrates the following "ship rescue" example to show the requirements of making the global service-based system as an AS^3 system.

There is a broken ship (BShip) with fifty passengers on board. Two nearby ports, P1 and P2, are located in two different countries, and there is a hospital located in each port. Two rescue ships, RShipA and RShipB, are at P1 and P2 respectively. Both rescue ships have enough capacity to hold fifty passengers, but only RShipB has a helicopter on-board (RHelicop). A rescue center service (RCenter) is responsible for coordinating rescue effort. Upon detecting the situation "An SOS signal is detected from a broken ship", RCenter is automatically invoked to receive location information as well as the status of passengers from BShip. At first, there is no injury report about the passengers on BShip. The RCenter dynamically discovers the nearby rescue units (RShipA and RShipB), and selects RShipA to perform the rescue mission since BShip is closer to P1. However, another report from BShip comes later indicating several life-threatening injuries caused by an explosion just happened on BShip. RShipA is unable to reach BShip and return to P1 fast enough to save the injured passengers' lives. Hence, RCenter finds RShipB with RHelicop on-board to rescue the injured passengers from BShip.

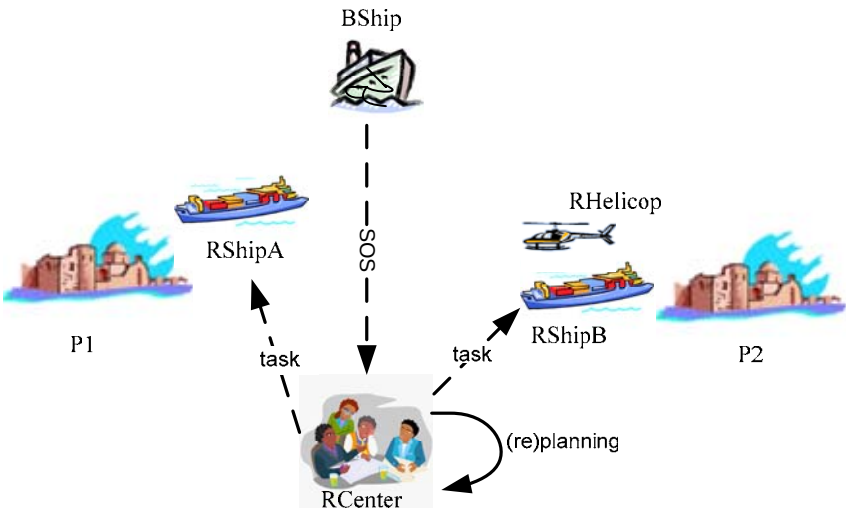

Figure 1. A Rescue Ship Example

From this example, we can generalize the following six inter-dependent requirements of developing, deploying and operating AS^3 systems:

R1). *Specification of AS^3 system*. The capabilities to precisely or formally specify various entities, situations, and security policies in an AS^3 system are the basis for developing AS^3 systems. For the above "rescue ship" example, services, such as RShipA, RShipB (which has the capability to call

another service, RHelicop), RHelicop and RCenter, need to be specified. To enable RCenter to plan an effective workflow to rescue BShip, the system need to consider contexts, such as locations of BShip, P1, P2, RShipA and RShipB, number of injured passengers on BShip, speed of RShipA, RShipB and RHelicop, and maximum flying distance of RHelicop. Based on these contexts, various situations, such as "*RCenter detects a broken ship that needs help*", "*there is no injury on the broken ship*", and "*there are life-threatening injuries on the broken ship*" need to be specified for the system to automatically recognize and response these situations. To ensure the security of services, various security policies need to be specified for the system to enforce. For example, since both the distance between P1 and BShip and the distance between P2 and BShip are greater than the flight radius of RHelicop (i.e., RHelicop cannot fly to BShip and return to P1 or P2 without refueling), we need to specify a policy like "*RHelicop can only leave RShipB until RShipB is close enough to BShip*".

R2). *Verification of AS^3 System.* Techniques and tools are needed to verify the specifications of the AS^3 system. The following properties of the specifications for AS^3 systems need to be verified:

- The consistency of situation specification, i.e. a situation cannot be recognized as true and false simultaneously.
- The consistency of security policy specification. For example, both permitted access and denied access to a service can be resulted from a set of policies. If this happen, no access will be denied by the specified policy for the service.
- The availability of the system. We need make sure the specified policies for an AS^3 system should not grant no authorization at all. For example no permission to access a service for every principal. This will make the service unavailable to everybody.
- Realizability of executing a workflow satisfying situation-awareness and security policies need to be analyzed.

R3). *Adaptive workflow planning*. AS^3 systems need to dynamically compose distributed services into processes based on mission goals of users, situations and relevant security policies and to adapt a process so that it can still achieve the mission goal of users when unexpected situations (e.g., services fail, security policies change) are detected. For example, the RCenter needs to dynamically discover the nearby rescue units (RShipA and RShipB), and plan a workflow to perform the rescue operations. When situation changes, RCenter needs to adaptively find other rescue units and re-plan or modify the workflow to rescue passengers on BShip.

R4). *Situation-awareness*. AS^3 systems need to have the capabilities to recognize situations, invoke appropriate services and enforce relevant security policies when situations change. For the above "rescue ship" example, the system needs to recognize the specified situations and response promptly to the situation changes. For example, when the situation of "*there are life-threatening injuries on the broken ship*" becomes true, RCenter needs to do re-planning to find RShipB with RHelicop on-board to rescue the injured passengers from BShip. To enforce security policies, situation-awareness is often desired [Yau05ab]. For example, in the policy of "*RHelicop can only leave RShipB until RShipB is close enough to BShip*", the situation of "*RShipB is close enough to BShip*" need to be recognized to ensure the success of the rescue operations and the security of RHelicop.

R5). *Situation-aware and secure workflow scheduling and execution*. AS^3 systems need to schedule and execute a process composed by distributed services on a given set of resources without violating the relevant security policies. For example, the workflow planned by the RCenter should be scheduled and executed efficiently and securely using the available resources for each service under current situation.

R6). *Distributed security policy enforcement.* AS3 systems need to enforce distributed security policies that are specified by different service providers and users. For example, the provider of RHelicop requires to enforce a security policy like "RHelicop can only leave RShipB until RShipB is close

enough to BShip" to ensure the security and the efficiency of RHelicop. The administrator of port P1 may require that "RHelicop can only land on P1 when there are life-threatening injuries need to be rescued by RHelicop". The RCenter may require that "Only RCenter can access RShipA and RShipB during when there is broken ship detected by the system. As services are dynamically composed into processes, it is necessary to enforce these distributed security policies during the execution of these processes to protect the security of AS^3 systems.

So far, this no effective enabling techniques so far available for systematically integrating all these requirements together to rapidly build AS^3 Systems, although various techniques have been proposed to address some these requirements individually such as security for Web services [WSS, WSSP, SAML, XACML] and composition of web services [Att96, BPEL4WS]. We discuss the related work in detail in the following section.

3. Current State of the Art

Currently, much research has been done in the following five aspects that are related to address some of the requirements described in Section 2.

3.1 Web Service Specification

(WSDL, OWL-S, to be added)

3.2 Service Contraction and Composition

Several methodologies have been developed to model processes of Web services [Gun93, Att96, Sch00]. In earlier work [Dav98] showed that Concurrent Transaction Logic (abbr., CTR) [Bon96] is a natural logical formalism for representing workflow control graphs, for reasoning with temporal and causality constraints on workflow execution, and for scheduling workflows in the presence of those constraints. Recently, we hasve substantially expanded this work to allow modeling of workflows to include non-trustworthy or adversarial services with the introduction of CTR-S [Dav04]. CTR-S is in the intersection of the areas of contracting and multi-party workflow modeling, and provides a formal framework for designing, modeling, and reasoning about the run-time properties of workflows made up of a collection of controllable and adversarial services.

~~However, a distinctive aspect of contracting in Web services, which is not captured by these methodologies, is that contracting involves multi-party processes, which can be adversarial or unreliable in nature.~~ ***[Davulcu: expand the current state of the art on adaptive workflow planning here]***

3.3 Situation-Awareness

[Dazhi: expand the current state of the art on situation-awareness here]

Substantial research has been done on human-computer interaction [End00], and information fusion [Jam01]. Most of these research efforts are focused on SAW modeling and situation analysis without considering development support for situation-aware applications.

3.4 Security Policy Specification and Analysis

One important aspect in distributed security management is how to specify security policies representing the security requirements of users. Industrial standards have been proposed to specifying security policies for Web services, such as WS-Security [WSS], WS-SecurityPolicy [WSSP], Security Assertion Markup Language (SAML) [SAML], eXtensible Access Control Markup Language (XACML) [XACML] and Secure Operations Language (SOL) [Bha02a]. WS-Security provides an XML framework for exchanging authentication and authorization information to secure the interaction among Web services and service invokers. WS-SecurityPolicy is used to describe the security policies in terms of their characteristics and supported features, such as required encryption algorithms, privacy rules, etc. SAML has the same purpose as WS-Security, but requires a third-party or services themselves to gather the evidence needed for policy decision. XACML is an XML framework for specifying access control policies for Web-based resources and can potentially be applied to secure service-based systems. SOL is a synchronous programming language for reactive systems, providing automatic verification of security policies.

Although, XML-based specification languages provide powerful expressiveness, these languages have no support for unambiguous policy semantics and mathematical analysis on security policies. Logic-based policy specification approaches have gained more interests for their unambiguous semantics, flexible

expression formats and various types of reasoning support. Default logic is a very flexible policy specification language that incorporates the Bell-LaPadula model and the complexity of decision computation is in quadratic time [Rei80]. Temporal Authorization Bases (TABs) [Ber00] labels each authorization policy with a periodic temporal expression. Although with periodic constraints TABs is decidable, it becomes undecidable if we add more flexible temporal constructs because time instance is unbounded. Flexible Authorization Framework [Jaj01] introduces the idea of hierarchical structure and allows users to specify how to resolve policy conflicts. Description logic [Baa03] is used to specify security policies because of its clean structure and powerful reasoning support. However, in order to make the reasoning possible and bounded, many guards are introduced, e.g. “equal” relation cannot be specified explicitly. Because of these guards, the expressions cannot be written in an intuitive way, which makes writing those expressions difficult. Note that none of these approaches can systematically address situation-awareness in the distributed security enforcement, which are very important for AS3 systems.

It is essential to analyze and verify the correctness of policy specification before enforcing them. Things become more complicated when operations span multiple organizations since the policies of each organization may be different or even contradictory to each other. Thus these organizations must have their policies resolved before any collaboration can happen. A security architecture based on separation kernel was introduced in [Gre03]. The policy clearly defines the basic separation axioms of non-exfiltration and non-infiltration for policy analysis. But other concepts involving definitions of state and the active partition are not clear. Moreover, the separation kernel implementation needs hardware support which may not always be the case. A formal framework for policy reconciliation was introduced in [Wan04] based on a directed acyclic graph model. A reconciliation algorithm was presented with consideration on participant preferences. But authorization and delegation are not addressed.

3.5 Distributed Security Policy Enforcement

Much research has been done on distributed security policy enforcement. PolicyMaker and KeyNote [Bla96, Bla99] provide a policy enforcement framework, which take local policies, received credentials,

and action strings as input, and determine whether the request should be granted or denied. However, PolicyMaker and KeyNote have no support for policy conflict analysis, policy composition, credentials discovery, and negative assertions. REFEREE (Rule-controlled Environment For Evaluation of Rules, and Everything Else) [Chu97] was developed to address policy conflictions. REFEREE can place all dangerous operations under policy control rather than making arbitrary decisions. However, Policy analysis support is not provided in REFEREE. Li et al [Li03] developed a role-based trust management system, called RT, based on the semantics of constraint Datalog. RT supports distributed credential discovery, which can process access requests with an incomplete set of credentials. However, RT only considers access control constraints, and provides no support for other security aspects in distributed systems.

4. Our Approach

In this section, we will first present our conceptual view of AS^3 systems, and then discuss our approach to developing, deploying and operating AS^3 systems, which will address the six requirements presented in Section 2.

4.1 The Conceptual View of AS^3 Systems

Figure 2 shows the top level conceptual view of our AS^3 systems. An AS^3 system is a collection of entities, acting in response to certain situations, and the interactions among the entities restricted by security policies. The various entities in an AS^3 system include services, users, processes, and computation/communication (comp/comm) resources. Users use processes to achieve their mission goals. A process may be composed by a set of services or implemented by a single service. The usage of services will consume certain amount of comp/comm resources. Each entity relates to some situations. For example, a service may be invoked under certain situations and change the current situation after it is invoked; a process may contain different execution paths that need to be selected in different situations; a

mission goal that a user wants to achieve is a desired situation in the AS^3 system that the user wants to have by using a process; the status of services or comp/comm resources may be captured by certain situations of the AS^3 system. Each entity may have its own security policies. For example, a service or comp/comm resource may have security policies that restrict the access to the service or the resource, and specify the obligations for using the service or the resource; a process may have security policies that control the delegations among the user and services involved in the process; a user may have security policies that defines what authorities she may have. Security policies in an AS^3 system are situation-aware, i.e., different security policy may need to be enforced under different situations.

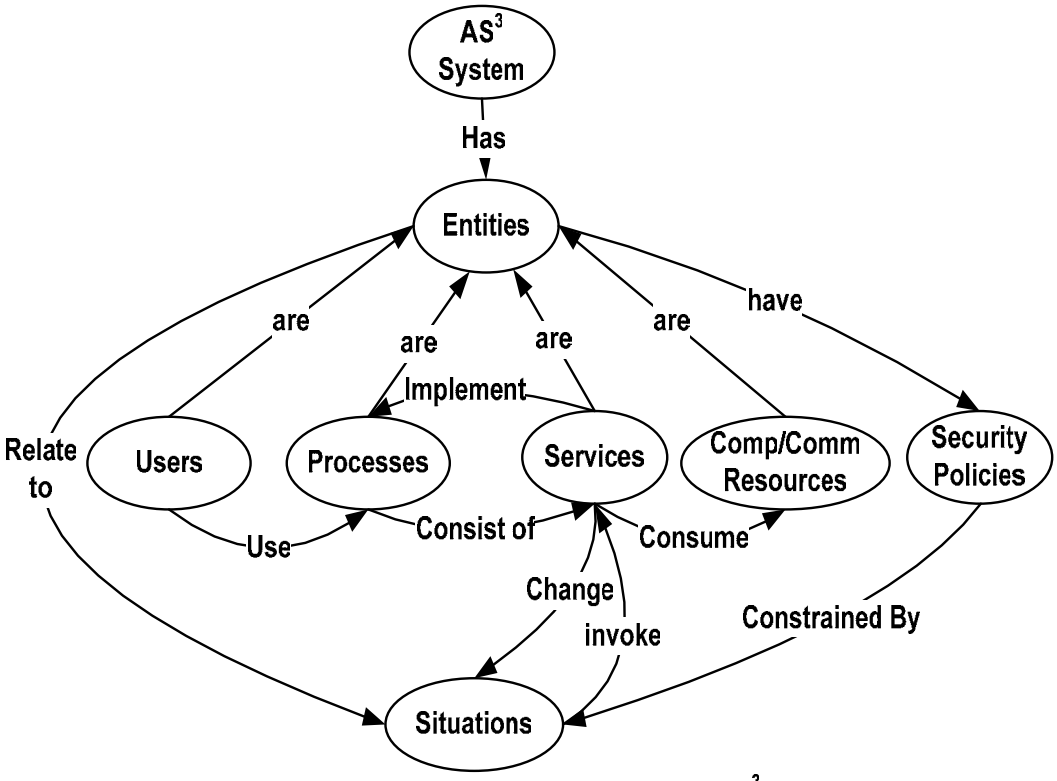

Figure 2. The Top Level Conceptual View of an AS^3 System

4.2 Our Approach to Developing, Deploying and Operating AS^3 systems

Our overall approach consists of the following four steps:

(1) To address requirement **R1,** we develop a formal model based on the conceptual view of AS^3 systems discussed in Section 4.1, and a formal specification language for AS^3 systems based on the developed formal model. Based on our conceptual view of AS^3 systems, the following primitives need to be provided in the formal model:

   i. Primitives for describing situation-awareness, including the primitives for defining contexts, situations and their relations to future actions.

   ii. Primitives for describing service semantics. These primitives are needed for adaptive workflow planning.

   iii. Primitives for defining security policies, including the primitives for defining authentication, authorization and delegation policies.

   To make the formal model of practical for AS^3 system, the following issues need to be considered:

   a) *Usability of the model*. The formal model must be easily understood and used by various service-providers or developers of AS^3 systems.

   b) *Tractability of the model*. The formal model must be tractable enough to allow interesting queries about service semantics, situation changes, security policies, to be answered efficiently.

   c) *Expressiveness of the model*. The formal model should expressiveness enough to support situation-awareness, service semantics and security policies.

   Currently, we are developing this formal model based on ambient logic. So far, we have extended the ambient logic to formally describe (1) the service semantics for workflow planning, and (2) access control policies for enforcing security. We are continuing to work on modeling situation-awareness and other types of security policies, such as authentication policies, delegation policies and auditing policies. In addition, the soundness and completeness of the formal model will be provided.

(2) To address requirement **R2**, we develop formal methods for verifying the consistency, completeness and realizability of specifications for AS^3 systems. In addition, we also develop other tools for specification analysis, such as suspension analysis.

(3) To address requirements **R3-6,** we develop the following techniques based on the formal model developed in (1):

a) Adaptive workflow planning (Requirement **R3**)

[Davulcu: add an overview for this] In AS^3 systems, services workflows may fail due to execution time problems; which may be due to incomplete domain information or due to unpredictable situation changes in external environment. Since classical planners [RusNor03] assume that goals and environment remain static, the initial services composition plans for achieving goals may fail during run-time. Various techniques based on plan repair [Krogt04] and re-planning[Krogt03] as well as techniques to deal with incomplete action specifications [Garland02] have been proposed to deal with problems related to plan adaptation. In AS^3 systems, we are developing domain-specific workflow adaptation techniques; where we provide (semi)-automated algorithms for mining successful and unsuccessful execution traces of workflows in order to identify and compose combinations of tasks into high-level tasks in order to engineer domain specific hierarchies of tasks. Such hierarchies may contain tasks such as "move a rescue ship within the range of the broken ship" or "perform all rescue operations on the broken ship". Online planning with high level tasks, where we plan some and then execute some, enables the synthesis of adaptive workflows that can take advantage of domain specific knowledge as the situation evolves during the execution.

Candidates for domain specific high level tasks can be identified by mining workflow execution traces using a sequence mining algorithm such as SPADE [Zaki01]. In the above "ship rescue" example, the mining algorithm may detect that, frequently a rescue ship needs the assistance of a satellite to track the location of a broken ship, hence the algorithm can suggest to group the satellite service and rescue ship into a high level task of "move a rescue ship within the range of the broken ship" upon approval of a domain expert. The next step involves determining the *preconditions* and *effects* of a new high level task. We are exploring the use of inductive logic programming [ILP94] techniques for learning the definitions of the precondition and effect predicates by mining the workflow execution log itself hence incorporating all the situation information as well as success and failure patterns into these predicate specifications.

Another issue arises when we synthesize a plan made from high level actions. As demonstrated in the "ship rescue" example, a planner may generate the plan with two steps: "move a rescue ship within the range of the broken ship" THEN "perform all rescue operations on the broken ship". An online planner might first select only one rescue ship, RShipA, during the planning for the "move a rescue ship within the range of the broken ship" based on the current situation information. However, during the movement of RShipA, if the situation changes such that "several life-threatening injuries caused by an explosion", an online planner for the second high level task "perform all rescue operations on the broken ship" can quickly synthesize a plan that incorporates RShipB as well as RShipA into the rescue operation – thus adapting the original plan for the emerging new situation.

b) Agent-based situation-aware adaptable service coordination. (Requirement **R4**)

[Dazhi: add an overview for this]

c) Distributed situation-aware workflow scheduling (Requirement **R5**)

[Dazhi: add an overview for this]

d) Agent-based distributed security policy management (Requirement **R6**). Security policies may be specified by different users in different security domains. Each security domain can have one or more security policy repositories. The specified security policies of each security domain are stored in its corresponding security policy repository. To improve the enforcement performance, we use distributed agents to enforce all these distributed security policies in service-based systems. It includes the following three parts: (i) Decompose security policies into different sets of security policies by analyzing the targets of each policy (the targets of a policy are determined by the entities in the policy); (ii) Synthesize the decomposed security policies to generate security agents; (iii) Deploy the security agents on an agent platform; (iv) discover and activate related security agents to evaluate and enforce security policies at runtime. During the enforcement of security policies, the security agents may need to communicate with situation-awareness agents to enforce situation-aware security policies.

The formal model and the formal specification language for AS^3 systems (step (1)) can enable the development of verification methods for AS^3 systems (step (2)), enable the semi-automated or automated system development, and provide the foundations for developing various enabling techniques for AS^3 systems (step (3)). ***Figure 3*** illustrates the high-level architecture of an AS^3 system. It integrates the components implementing the techniques developed in step (3) with the protocols that determine the way a component interacts with others. The services are used for publishing various capabilities provided by organizations. The specification language that we are currently developing will be used to describe the semantics, and the related situations and security policies of services in AS^3 systems. Similar to normal service-based systems, the description of services will be published in the *Directories*. The following agents and services are being developed to provide runtime support for SAW, security and adaptability in AS^3 systems:

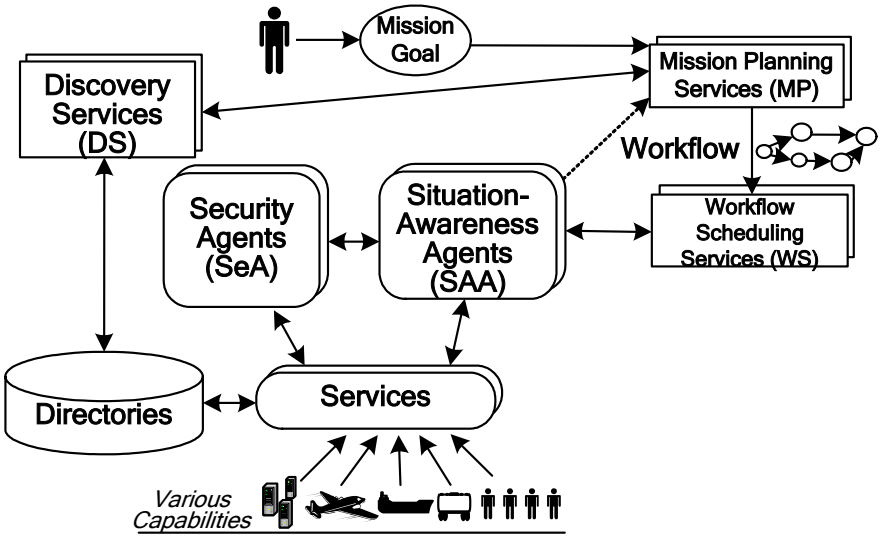

Figure 2. A High-level Architecture of AS^3 Systems

- *SAW Agents* (SAA) that collect context data of interest, analyze the collected context data to determine the situation, execute workflows and adapt workflows when relevant situation changes occur;
- *Security Agents* (SeA) that discover and enforce relevant security policies in a distributed manner;
- *Discovery Services* (DS) that perform semantic-based situation-aware service discovery;

- *Mission Planning Services* (MP) that generate workflow to fulfill certain mission goals based on security policies, situations and available services; and
- *Workflow Scheduling Services* (WS) that decompose the generated workflows, and generate, deploy and initialize SAAs and SeAs to execute workflows in such a way that all relevant security policies and resource constraints are satisfied.

An AS^3 system is operated as follows:

(1) A user (either a human user or an application) specifies the mission goal and additional security or situational constraints that need to be satisfied when achieving the mission goal, and sends the information to a MP.

(2) The MP will first use available DSs to identify services that might be needed to achieve the mission goal, and then start the mission planning process to generate a workflow that can achieve the specified mission goal using available services with all relevant security policies and situational constraints satisfied.

(3) The generated workflow will then be decomposed by a WS to generate SeAs and SAAs for executing the workflow and enforcing security policies.

(4) During the execution of the workflow, the SAAs can adapt the workflow by selecting alternative services or invoke the MP to perform partial or complete re-planning on the occurrence of certain situation changes that make the execution of the workflow violate the situational or security constraints of some services used in the workflow.

(5) For the partial re-planning, the SAAs will notify the MP of the completed portion of the initial workflow, and the MP will try to generate a new workflow to achieve the specified mission goal based on the partial result generated from the completed portion of the initial workflow. In case such a new workflow cannot be found, the MP will re-plan the entire workflow without using the services

that cause the failure. However, there is no guarantee that a new workflow can be found without using the failed services.

5. Applying Our Approach in the "Ship Rescue" Example

[To be added soon]

6. Conclusion and Future Work

In this paper, we give an overview of our approach to the rapid development, deployment and operation of Adaptable Situation-aware Secure Service-based Systems. Several important research issues are identified, and our ideas for developing new techniques to address these issues are briefly discussed. Currently, we are developing these techniques and a demonstration system based on Secure Infrastructure for Networked Systems (SINS) [References???] developed by Dr. Bharadwaj in Naval Research Laboratory (NRL).

Acknowledgement

This work is supported by the Office of Naval Research under the Multidisciplinary Research Program of the University Research Initiative, Contract No. N00014-04-1-0723.

References

[Att96] P.C. Attie, M.P. Singh, E.A. Emerson, A.P. Sheth, M. Rusinkiewicz, "Scheduling Workflows by Enforcing Intertask Dependencies," *Distributed Systems Engineering J.*, vol. 3 no.4, 1996, pp. 222-238

[Baa03] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider, editors, "The Description Logic Handbook," *Cambridge University Press* 2003

[Ber00] E. Bertino, P. A. Bonatti, E. Ferrari and M. L. Sapino, "Temporal Authorization Bases: From Specification to Integration," *Journal of Computer Security*, 8(4), 2000

[Bha02a] R. Bharadwaj, "SOL: A Verifiable Synchronous Language for Reactive Systems," *Electronic Notes in Theoretical Computer Science*, vol. 65, no. 5, 2002.

[Bha02b] R. Bharadwaj, "Verifiable Middleware for Secure Agent Interoperability," *Proc. 2nd IEEE W. on Formal Approaches to Agent-Based Systems (FAABS II)*, 2002, pp. 126-132

[Bla96] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," *IEEE Conference on Privacy and Security*, Oakland, 1996.

[Bla99] M. Blaze, et al., "The KeyNote Trust Management System (version 2)," *RFC2704*, 1999

[D8100.1] U.S. Department of Defense Directive (DODD) 8100.1: "Global Information Grid (GIG) Overarching Policy," The Pentagon, Washington D.C., September 2002.

[Dav98] H. Davulcu, M. Kifer, C.R.Ramakrishnan, I.V.Ramakrishnan. Logic based modeling and analysis of workflows. *ACM Symposium on Principles of Database Systems (PODS)*, 1998.

[Dav04] H. Davulcu, M. Kifer and I.V. Ramakrishnan, "CTR-S: A Logic for Specifying Contracts in Semantic Web Services", *Proc. 13th Int'l World Wide Web Conf.*, 2004.

[End00] M.R. Endsley, "Theoretical Underpinnings of Situation Awareness: A Critical Review", *Situation Awareness Analysis and Measurement*, Lawrence Erlbaum Associates, 2000, pp.3-32.

[Garland02] A.Garland and N. Lesh, Continuous Plan Evaluation with Incomplete Action Descriptions, *Proc. 3rd Int'l NASA WS on Planning and Scheduling for Space*, Houston, TX, 2002.

[Gre03] D. Greve, M. Wilding, and W.M. Vanfleet, "A Separation Kernel Formal Security Policy", *in Proc. Of the ACL2 Workshop 2003*, July 2003

[Gun93] R. Gunthor, "Extended transaction processing based on dependency rules", *Proc. RIDE-IMS Workshop*, 1993.

[ILP94] N.Lavrac and S. Dzeroski. "Inductive Logic Programming: Techniques and Applications", Ellis Horwood, New York, 1994.

[Jaj01] S. Jajodia, Pamarati. S, M.L. Sapino, and V.S. Subrahmanian, "Flexible Supporting for Multiple Access Control Policies," *ACM Trans. on Database Systems*, 26(2):214-260, 2001

[Jam01] S.M. Jameson, "Architecture for Distributed Information Fusion to Support Situation Awareness on the Digital Battlefield", *Proc. of 4th Int'l Conf. on Data Fusion*, August 2001.

[Krogt04] R. van der Krogt and M. de Weerdt. The two faces of plan repair. In *Proceedings of the Sixteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC-04)*, pages 147-154, 2004.

[Krogt03] R. van der Krogt, M. de Weerdt, and C. Witteveen. A resource based framework for planning and replanning. *Web Intelligence and Agent Systems*, 1(3/4):173-186, 2003.

[Li03] N. Li and J. Mitchell. "RT: A Role-Based Trust Management Framework," *Proc. 3rd DARPA Information Survivability Conf. and Exposition* (DISCEX '03), Apr. 2003.

[Rei80] R. Reiter, "A Logic for Default Reasoning," *Artificial Intelligence*, 13:81-132, 1980

[RusNor03] P. Norvig S. Russell. *Artificial Intelligence: A Modern Approach*, chapter 13, pages 392-412. Parentice Hall, New Jersey, 1 edition, 1995.

[Sch00] C. Schlenoff et al., "The Essence of the Process Specification Language," *Trans. of the Society for Computer Simulation Int'l*, vol. 16(4), 2000, pp. 204-216.

[SAML] OASIS Security Services TC, "Security Assertion Markup Language (SAML)," URL: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[Wan04] H. Wang, S. Jha, M. Livny, and P.D. McDaniel, "Security Policy Reconciliation in Dsitributed Computing Environments", *IEEE 5th International Workshop on Policies for Distributed Systems and Networks*, June 7-9, 2004

[WSA04] Web Services Architecture. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[WSS] WS-Security. http://www-106.ibm.com/developerworks/webservices/library/ws-secure/.

[WSSP] WS Security Policy. http://www-106.ibm.com/developerworks/library/ws-secpol/.

[XACML] OASIS eXtensible Access Control Markup Language (XACML) TC, URL: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[Yau02a] S. S. Yau, Y. Wang and F. Karim, "Development of Situation-Aware Application Software for Ubiquitous Computing Environments," *Proc. 26th Ann. Int'l Computer Software and Applications Conference (COMPSAC 2002)*, 2002, pp. 233-238.

[Yau02b] S. S. Yau et al., "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," *IEEE Pervasive Computing*, vol. 1(3), 2002, pp. 33-40.

[Yau05a] S. S. Yau, Y. Yao and V. Banga, "Situation-Aware Access Control for Service-Oriented Autonomous Decentralized Systems," *7th Int'l Symposium on Autonomous Decentralized Systems*, 2005, to appear

[Yau05b] S. S. Yau, Y. Yao, Z. Chen and L. Zhu, "An Adaptable Security Framework for Service-based Systems," 10th IEEE Int'l Workshop on Object-oriented Real-time Dependable Systems (*WORDS2005*), to appear

[Zaki01] M. J. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, 2001.