# Adaptable Situation-Aware Secure Service-Based ($AS^3$) Systems

[1]S. S. Yau, [1]H. Davulcu, [2]S. Mukhopadhyay, [1]D. Huang, and [1]Y. Yao
[1]*Arizona State University, Tempe, AZ*
[2]*West Virginia University, Morgantown, WV*
[1]*{yau, hasan.davulcu, dazhi.huang, yisheng.yao}@asu.edu*
[2]*supratik.mukhophadyay@mail.wvu.edu*

## Abstract

*Service-oriented systems are distributed systems which have the major advantage 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 systems, various capabilities are provided by different organizations as services interconnected by various types of networks. The services can be integrated following a specific workflow 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, anytime with 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, an approach to rapid development of adaptable situation-aware secure service-based ($AS^3$) systems is presented. Our approach enables users to rapidly generate, discover, compose services into processes to achieve their goals based on the situation and adapt these processes when situation changes.*

**Keywords**: Service-based systems, adaptive workflow planning, situation-awareness, distributed trust management, specification language, verification.

## 1. Introduction

Service-based systems are distributed computing systems which have the major advantage of enabling rapid composition of distributed applications, regardless of programming languages and platforms used in developing and running different components of the applications. These systems are being adopted in many large-scale distributed systems, such as Grid and Global Information Grid (GIG) [2], for various applications including collaborative scientific and engineering work, e-business, healthcare, military, and homeland security. In service-based systems, various capabilities 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 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 to PCs), knowing that their confidentiality and privacy are well protected and the systems will adapt to their needs in various 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 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 [3, 4]. A *situation* is a set of context attributes over a period of time that is relevant to future system behavior [3, 4]. A *context* is any instantaneous, detectable, and relevant property of the environment, the system or users, such as time, location, wind velocity, temperature, available bandwidth, invocation of action, and a user's schedule [3, 4]. SAW is essential for high confidence and adaptable service-based systems since 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 [5-9], so far there are no effective enabling techniques for developing *Adaptable Situation-aware Secure Service-based* ($AS^3$) *Systems*. In this paper, we will first discuss the adaptability, SAW, and security requirements of service-based systems and outline our approach to achieving the two objectives: (1) Enable rapid development of and provide runtime support for *hierarchical SAW* and *distributed security policy management* in $AS^3$ systems. (2) Enable users to rapidly discover, contract with and compose reliable and unreliable services into adaptive processes to achieve their goals based on the situation. An example will be given to illustrate the requirements of $AS^3$ systems and the rationale of our approach.

## 2. Rationale and Requirements

### 2.1. Rationale of $AS^3$ Systems

Consider a typical global service-based system which 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. The following "ship rescue" example using such a global service-based system shows that the requirements of such an application naturally can be satisfied by an $AS^3$ system.

Consider the following example: There is a broken passenger ship (**BS**) with 200 people on board. Two nearby ports, **P1** and **P2**, are in two different countries, and there is a hospital in each port. Two rescue ships, **A** and **B**, are at **P1** and **P2** respectively. Both rescue ships have enough capacity to hold 200 people, but only **B** has a helicopter **H** on-board. A rescue center **RC** is responsible for coordinating the rescue operations. The following services are involved in this example:

- $\mathbf{S_{BS}}$: the service on **BS** for sending out SOS signal with various context data, including the number of life-threatening injuries.
- $\mathbf{S_H}$: the rescue service for controlling the helicopter **H** on **B** to perform rescue operations.
- $\mathbf{S_A}$: the rescue service for controlling **A** to perform rescue operations.
- $\mathbf{S_B}$: the rescue service for controlling **B** to perform rescue operations. $\mathbf{S_B}$ may invoke $\mathbf{S_H}$.
- $\mathbf{S_{RC}}$: the service at the rescue center **RC** for receiving and analyzing information from other services, and coordinating rescue operations.

Upon detecting the situation "*An SOS signal is detected from ship **BS***", $\mathbf{S_{RC}}$ automatically generate a plan to perform the necessary rescue operations for **BS**. At the beginning, there is no injury report about the people from **BS**. The $\mathbf{S_{RC}}$ dynamically discovers nearby rescue services ($\mathbf{S_A}$ and $\mathbf{S_B}$), and selects $\mathbf{S_A}$ to perform the rescue mission since **BS** is closer to P1. However, after **A** is on the way to **BS**, another report from **BS** is received reporting that several people on BS have life-threatening injuries caused by an explosion just happened on BS. **A** is unable to reach BS and return to P1 fast enough to save the injured people on BS. Hence, $\mathbf{S_{RC}}$ finds $\mathbf{S_B}$ and $\mathbf{S_H}$ to rescue the injured people from **BS**.

To develop the above example application on a global service-based system for ensuring success in the rescue operations, the following requirements need to be satisfied:

- *Adaptability*. The system needs to dynamically discover the nearby rescue services ($\mathbf{S_A}$ and $\mathbf{S_B}$), and plan a workflow to perform the rescue operations. When situation changes, the system needs to adaptively find other rescue services and re-plan or modify the workflow to rescue people on **BS**.
- *Context-acquisition.* The system needs to consider the following contexts:
  - The locations of **A**, **B**, **BS**, **P1** and **P2**
  - The number of injured people on **BS**
  - The speed of each of **A**, **B** and **H**
  - The maximum flying distance of **H**
- *SAW*. The system needs to automatically recognize and respond to the following situations based on the acquired context data:
  - $\mathbf{S_{RC}}$ detects a broken ship that needs help
  - There is no injury on **BS**
  - There are life-threatening injuries on **BS**
- *Security*. For the security of services, the system needs to enforce the following security policies:
  - Since the distances between **P1** and **BS** and between **P2** and **BS** are greater than the flight radius of **H** (i.e., **H** cannot fly to **BS** and return to **P1** or **P2** without refueling), $\mathbf{S_H}$ can be called for rescuing **BS** only when **B** is close enough to **BS**.
  - $\mathbf{S_A}$ and $\mathbf{S_B}$ can only be called by $\mathbf{S_{RC}}$.

Although various techniques have been proposed to address some of these requirements individually, such as security for Web services [6-9] and composition of web services [5] (see Section 3 for details), so far there is no effective enabling technique to systematically build such a service-based system rapidly to satisfy all these requirements. The objective of the *Adaptable*

*Situation-aware Secure Service-based* ($AS^3$) *Systems* is to naturally satisfy all these requirements.

### 2.2. Requirements for Building $AS^3$ Systems

To enable rapid development of $AS^3$ systems, we will need the following support:

**R1)** *Support for formally specifying the semantics of services, and requirements for SAW and security.*

**R2)** *Support for verifying workflow, SAW and security specifications.* Tools and techniques are needed for verifying the following properties of the workflow, SAW and security specifications: (i) The consistency of specifications for SAW and security policy. (ii) The accessibility of the system, i.e. the specified policies for an $AS^3$ system should not deny all possible access requests. (iii) The execution of a workflow satisfies all SAW and security requirements.

**R3)** *Adaptive workflow planning*. $AS^3$ systems need to dynamically compose distributed services into processes based on the goals of users, situations and relevant security policies and to adapt a process so that it can still achieve the users' goals when unexpected situations (e.g., services fail, security policies change) are detected.

For example, the $\mathbf{S_{RC}}$ needs to dynamically discover the nearby rescue services ($\mathbf{S_A}$ and $\mathbf{S_B}$), and plan a workflow to perform the rescue operations. When situation changes, $\mathbf{S_{RC}}$ needs to adaptively find other rescue units and re-plan or modify the workflow to rescue people on ship **BS**.

**R4)** *Distributed situation-aware and secure workflow scheduling*. $AS^3$ systems need to decompose workflows composed by distributed services into distributed processes and schedule these processes on a given set of resources efficiently without violating the relevant security policies.

In $AS^3$ systems, multiple workflows may be generated together and the execution process of a generated workflow may overlap with others. Since invoking services during workflow execution will cause some situation changes, which will change the security policies to be enforced, the execution of the generated workflows must be properly scheduled so that all of them can be executed with all dependencies on situations and security policies of these workflows satisfied. Furthermore, the execution of the generated workflows must be scheduled to satisfy the timing requirements and utilize resources efficiently.

**R5)** *SAW for adaptable service coordination*.

In an $AS^3$ system, the distributed processes, each of which contains a sequence of service invocations, need to be coordinated to ensure correct execution of the workflow under varying situations. Hence, $AS^3$ 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 respond promptly to the situation changes. For example, when the situation of "*there are life-threatening injuries on ship **BS***" becomes true, $\mathbf{S_{RC}}$ needs to do re-planning to find $\mathbf{S_B}$ and $\mathbf{S_H}$ to rescue the injured people from **BS**. To enforce security policies, SAW is often desired [11, 12]. For example, in the policy of "$\boldsymbol{S_H}$ *can be called for rescuing **BS only** when **B** is close enough to **BS***", the situation of "***B** is close enough to **BS***" needs to be recognized to ensure the success of the rescue operations and the security of **H**.

**R6)** *Distributed security policy enforcement.* $AS^3$ systems need to enforce distributed security policies that are specified by different service providers and 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.

For example, the provider of $\boldsymbol{S_H}$ requires to enforce a security policy like "$\boldsymbol{S_H}$ *can be called for rescuing **BS** when **B** is close enough to **BS***" to ensure the security of $\mathbf{S_H}$. The $\mathbf{S_{RC}}$ may require that "*Only $\boldsymbol{S_{RC}}$ can access* $\mathbf{S_A}$ and $\mathbf{S_B}$ *when there is a 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.

## 3. Current State of the Art

Currently, much research has focused on the following five aspects related to some of the above requirements.

### 3.1. Web Service Specification

Several specification languages have been developed for specifying Web services, among which Web Services Description Language (WSDL) [13] and Ontology Web Language for Web Services (OWL-S) [14] are most popular. WSDL is an XML-formatted language for describing a Web service's capabilities as collections of communication endpoints capable of exchanging messages. WSDL provides a basic and simple abstraction of Web services. OWL-S provides primitives for service descriptions in semantic web. Its overall structure includes three main parts: the *service profile* for advertising and discovering services; the *process model*, which gives a detailed description of a service's operation; and the *grounding*, which provides details on how to interoperate with a service, via

messages. However, formalisms for expressing conditions, such as situational and security conditions are not defined in OWL-S.

### 3.2. Service Contraction and Composition

Several methodologies have been developed to model processes of Web services [10, 15, 16]. Earlier work [17] showed that Concurrent Transaction Logic (CTR) [18] 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 have substantially expanded this work to allow modeling of workflows to include non-trustworthy or adversarial services with the introduction of CTR-S [19]. 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.

### 3.3. SAW

SAW has been studied in artificial intelligence, human-computer interaction and data fusion communities [20-25]. In [20], a Situation Calculus was introduced for reasoning on various facts of a situation without explicitly specifying what constitutes a situation. Situation Calculus was later extended in [21] to support temporal reasoning in Situation Calculus. In [22], formal notation and semantics for situations were presented, and situation is considered as a part of reality that can be observed and reasoned about. In [23], a formal framework for SAW formalizing the data fusion process was presented. This formal framework can be expressed in various languages, including UML, DAML and the Slang methods language [23]. A core SAW ontology using OWL was presented in [24], and used to construct RuleML-based domain theories in [25]. However, most of these research efforts are focused on modeling SAW and situation analysis without considering the support to development of situation-aware systems.

### 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 [6], WS-SecurityPolicy [7], eXtensible Access Control Markup Language (XACML) [8], and Security Assertion Markup Language (SAML) [9]. 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 and privacy rules. 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. Although XML-based security policy specification languages provide powerful expressiveness, these languages have no support for formal analysis on security policies.

Logic-based policy specification languages have gained more interests for their unambiguous semantics, flexible expression formats and various types of reasoning support. Default logic [26] is a very flexible policy specification language that incorporates the Bell-LaPadula model [27] and the complexity of decision evaluation is in quadratic time. Temporal Authorization Bases (TABs) [28] labels each authorization policy with a periodic temporal expression. Although with periodic constraints TABs is still decidable, it becomes undecidable if we add more flexible temporal constructs since time instance is unbounded. Flexible Authorization Framework [29] introduces the idea of hierarchical structure and allows users to specify how to resolve policy conflicts. Description logic (DL) [30] is used to specify security policies for its clean structure and powerful reasoning support. However, to make the reasoning bounded, many restrictions are introduced, and hence security policy cannot be easily written in an intuitive way. Note that none of these approaches can address temporal constraints, situation-awareness and hierarchical structure at the same time. Note that none of these approaches can systematically address SAW in the distributed security enforcement, which is very important for $AS^3$ systems.

### 3.5. Distributed Security Policy Enforcement

Much research has been done on distributed trust management, which is related to our adaptable security framework. *PolicyMaker* [31] and *KeyNote* [32] provide a policy enforcement framework, which uses local policies, credentials, and action strings as input to determine whether the request should be granted or denied. However, *PolicyMaker* and *KeyNote* have no support for credentials discovery, negative assertions, policy analysis, and policy composition. *REFEREE*

[33] places all critical operations under policy control rather than making arbitrary decisions, which may cause certain dangerous operations to occur. In [34], a role-based trust management system, called *RT*, based on the semantics of constraint Datalog was presented. *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 aspects in a trust management system. In these systems, trust relationship solely depends on credential verification, and hence it is relatively static.

# 4. Our Approach

In this section, we will first present our conceptual view of $AS^3$ systems, and then present our approach to developing, deploying and operating $AS^3$ systems, to satisfy the six requirements discussed in Section 2.2.

## 4.1. The Conceptual View of $AS^3$ Systems

Figure 1 shows the top-level conceptual view of $AS^3$ 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 entities of an $AS^3$ system include services, users, processes, and computation/ communication (comp/comm) resources. Users use processes to achieve their goals. A process may be composed of a set of services or implemented by a single service. The usage of services will consume a certain amount of computing and communicating (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 users and services involved in the process; a user may have security policies that defines what authorities he/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.

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

Our overall approach to developing, deploying and operating $AS^3$ systems consists of the following three steps:

Step (1) *To satisfy **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 are provided in the formal model: (i) Primitives for describing SAW, 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 useful for $AS^3$ systems, 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 to allow interesting queries about service semantics, situation changes, and security policies, to be answered efficiently.

c) *Expressiveness of the model*. The formal model should have strong expressiveness sufficient to support SAW, service semantics and security policies.

In order to realize the models from the requirements of $AS^3$ systems automatically, we specify the requirements of an $AS^3$ system along with the QoS goals in an $AS^3$ logic, which is a decidable modal logic based on the ambient logic [35, 36]. So far we use this logic to formally describe the service theory, which includes input-output descriptions of the services, specification and recognition of situations, and access control policies for enforcing security. A model realizing these requirements can be synthesized from its proof from the service theory. We are continuing to work on modeling authentication policies, delegation policies and auditing policies.

Step (2) *To satisfy **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.

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

3a) Adaptive workflow planning (**R3**). In $AS^3$ systems, workflows may fail due to execution time

problems; which may be due to incomplete domain information or due to unpredictable situation changes. 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, and 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 [37]. In the above "ship rescue" example, the mining algorithm may detect that, a rescue ship frequently needs the assistance of a satellite to track the location of a broken ship, and 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 [38] techniques for learning the definitions of the precondition and effect predicates by mining the workflow execution log itself, and hence incorporating all the situation information as well as success and failure patterns into these predicate specifications. In addition, we are exploring techniques for on-line synthesis of a new plan for adapting the original plan for emerging new situations during the execution.

3b) Distributed situation-aware and secure workflow scheduling (**R4**). To schedule workflow with dependencies on situations and security policies efficiently, we are developing a distributed scheduling approach inspired by event algebra [39], Concurrent Constraint Transaction Logic (CCTR) [40] and GridFlow [41]. Our approach consists of the following major steps: (i) Identify required resources based on a generated workflow. (ii) Decompose the generated workflow into distributed processes, and compute the guards (conditions) for the execution of the distributed processes based on relevant security policies, situational constraints and timing constraints. If no feasible decomposition can be found, notify the planner to perform re-planning. (iii) Check whether any situational, security and timing constraints of other workflows already running in the system are violated if the distributed processes generated in (ii) are deployed and executed. (iv) If no constraint is violated, search for an optimal resource allocation for the distributed processes based on the resource utilization requirements and availability of the required comm/comp resources; otherwise, repeat (ii) with additional constraints generated from the distributed processes that cause the violations. (v) Deploy and execute the workflow.

3c) SAW agents for adaptable service coordination. (**R5**). Since it is very expensive (in terms of computation) to enumerate all possible situation changes during the planning and scheduling process, we use distributed SAW agents to adaptively coordinate the execution of a generated workflow to ensure that all situational dependencies of the workflow are satisfied (security agents for enforcing security policies will be discussed in 3d). A SAW agent works as follows: (i) Accept a workflow from the workflow scheduler or other agents; (ii) discover other SAW agents for cooperative situation analysis and/or service coordination if context and situation information and/or required services in the workflow are not available locally; (iii) Acquire contexts and analyze situations; (iv) Select and invoke the appropriate services monitored by the SAW agent based on the workflow and the current situation, and send partial workflow to other SAW agents if necessary; (v) Adapt the workflow by searching alternative services or performing re-planning when a dependency on situations cannot be satisfied.

3d) Agent-based distributed security policy management (**R6**). To improve the enforcement performance, we use distributed agents to enforce all these distributed security policies in service-based systems. Our approach works as follows: (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 SAW agents to enforce situation-aware security policies.

***Figure 2*** 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 $AS^3$ systems:

- *SAW Agents* (SAA) which collect context data of interest, analyze the collected data to determine the situation, execute workflows and adapt workflows when relevant situation changes occur.
- *Security Agents* (SeA) which discover and enforce relevant security policies in a distributed manner.
- *Discovery Services* (DS) which perform semantic-based situation-aware service discovery.
- *Mission Planning Services* (MP) which generate workflows to fulfill certain mission goals based on security policies, situations and available services.
- *Workflow Scheduling Services* (WS) which 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 operates as follows:

S1) 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 the AS3 system achieves the mission goal, and sends the information to the MP.

S2) The MP will first use available DSs to identify services that may be needed to achieve the mission goal, and then start the 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.

S3) The generated workflow will then be decomposed by a WS to generate SeAs and SAAs for executing the workflow and enforcing security policies.

S4) 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 to violate the situational or security constraints of some services used in the workflow.

S5) For the partial re-planning, the SAAs will notify the MP of the completed portion of the initial workflow, and the MP will generate a new workflow to achieve the 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. If this is the case, the user will be informed that the system is currently unable to provide the necessary services for achieving the user's mission goal and the user will need to modify the mission goal.

## 5. Conclusion

In this paper, we have presented the rationale and requirements of rapid development of $AS^3$ systems, and an overview of our approach to satisfying these requirements. Challenging research issues arising in our approach 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) [42] developed by US Naval Research Laboratory (NRL).

## Acknowledgment

This work is supported by the Department of Defense/Office of Naval Research under the Multidisciplinary Research Program of the University Research Initiative, Contract No. N00014-04-1-0723. We would like to thank Ramesh Bharadwaj of NRL for his constructive comments and many helpful discussions.

## References

[1] W3C, "Web Services Architecture," http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

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

[3] S. S. Yau, et al., "Development of Situation-Aware Application Software for Ubiquitous Computing Environments," *Proc. 26th Ann. Int'l Computer Software and Applications Conf. (COMPSAC 2002)*, 2002, pp. 233-238.

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

[5] Business Process Execution Language for Web Services. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

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

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

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

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

[10] P. C. Attie, et al., "Scheduling Workflows by Enforcing Intertask Dependencies," *Distributed Systems Engineering J.*, vol. 3(4), 1996, pp. 222-238.

[11] S. S. Yau, et al., "Situation-Aware Access Control for Service-Oriented Autonomous Decentralized Systems," *7th Int'l Symp. on Autonomous Decentralized Systems*, to appear.
[12] S. S. Yau, et al., "An Adaptable Security Framework for Service-based Systems," *10th IEEE Int'l Workshop on Object-oriented Real-time Dependable Systems*, to appear.
[13] W3C, "Web Services Description Language (WSDL) 1.1," http://www.w3.org/TR/wsdl
[14] W3C, "OWL-S: Semantic Markup for Web Services," http://www.daml.org/services/owl-s/1.1/overview/
[15] R. Gunthor, "Extended transaction processing based on dependency rules", *Proc. RIDE-IMS Workshop*, 1993, pp. 207-214.
[16] 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.
[17] H. Davulcu, et al., "Logic based modeling and analysis of workflows", *ACM Symp. on Principles of Database Systems (PODS)*, 1998, pp.25-33.
[18] A.J. Bonner and M. Kifer, "Concurrency and Communication in Transaction Logic", *Proc. Joint Int'l Conf. and Symp. on Logic Program*ming, 1996, pp. 142-156.
[19] H. Davulcu, et al., "CTR-S: A Logic for Specifying Contracts in Semantic Web Services", *Proc. 13th Int'l WWW Conf*., 2004, pp.144-153.
[20] J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence", *Machine Intelligence 4*, 1969, pp. 463-502.
[21] J. A. Pinto, "Temporal Reasoning in the Situation Calculus", PhD Thesis, University of Toronto, 1994.
[22] J. Barwise, "Scenes and Other Situations", *J. Philosophy*, vol. 77, 1981, pp. 369-397.
[23] K. Baclawski, et al., "Formalization of Situation Awareness," *Proc. 11th OOPSLA Workshop on Behavioral Semantics*, 2002, pp. 1-15.
[24] C. J. Matheus, et al., "A Core Ontology for Situation Awareness", *Proc. 6th Int'l Conf. on Information Fusion*, 2003, pp. 545 –552.
[25] C. J. Matheus, et al., "Constructing RuleML-Based Domain Theories on top of OWL Ontologies", *Proc. 2nd Int'l Workshop on Rules and Rule Markup Languages for the Semantic We*b, 2003, pp. 81–94.
[26] R. Reiter, "A Logic for Default Reasoning," *Artificial Intelligence*, vol. 13, 1980, pp.81-132.
[27] D. Bell and L. Lapadula, "Secure Computer System: Unified Exposition and Multics Interpretation." *Technical Report MTR-1997*, MITRE, 1976.
[28] E. Bertino, et al., "Temporal Authorization Bases: From Specification to Integration," *J. Computer Security*, vol. 8(4), 2000.
[29] S. Jajodia, et al., "Flexible Supporting for Multiple Access Control Policies," *ACM Trans. on Database Systems*, vol. 26(2), 2001, pp. 214-260.
[30] F. Baader, et al., editors, "The Description Logic Handbook," Cambridge University Press, 2003
[31] M. Blaze, et al., "Decentralized Trust Management," *Proc. IEEE Symposium on Privacy and Security*, Oakland, 1996, pp. 164-173.
[32] M. Blaze, et al., "The KeyNote Trust Management System (version 2)," *RFC2704*, 1999.
[33] Y. Chu, et al., "REFEREE: Trust Management for Web Applications." *World Wide Web J.*, vol.2(3),1997,pp.127-139.
[34] N. Li and J. Mitchell. "RT: A Role-Based Trust Management Framework," *Proc. 3rd DARPA Information Survivability Conf. and Exposition* (DISCEX '03), Apr. 2003,
[35] L. Cardelli and A .D. Gordon, "Anytime, Anywhere: Modal Logics for Mobile Ambients", *Proc. 27th ACM Symp. on Principles of Programming Languages*, 2000, pp 365-377.
[36] W. Charatonik, et al., "Model Checking Mobile Ambients", *Theoretical Computer Science*, vol. 308(1-3), 2003, pp. 277-331.
[37] M. J. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, 2001.
[38] N. Lavrac and S. Dzeroski. "Inductive Logic Programming: Techniques and Applications", Ellis Horwood, New York, 1994.
[39] Munindar P. Singh, et al., "An Event Algebra for Specifying and Scheduling Workflows," *Proc. 4th Int'l Conf. on Database Systems for Advanced Applications (DASFAA'95)*, 1995, pp. 53-60.
[40] P. Senkul, et al., "A Logical Framework for Scheduling Workflows under Resource Allocation Constraints," *Proc. 28th Int'l Conf. on Very Large Data Bases*, 2002, pp. 694-705.
[41] J. Cao, et al., "GridFlow: Workflow Management for Grid Computing," *Proc. 3rd IEEE/ACM Int'l Symp. On Cluster Computing and the Grid*, 2003, pp. 198-205.
[42] R. Bharadwaj, "A Framework for the Formal Analysis of Multi-Agent Systems," *Proc. Formal Approaches to Multi-Agent Systems (FAMAS)*, 2003.