Int J Software Informatics, Volume 7, Issue 1 (2013), pp. 21–39 Tel: +86-10-62661040
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org

Email: ijsi@iscas.ac.cn, ijsi2007@gmail.com

Development of Situation-Aware Applications in Services and Cloud Computing Environments

Stephen S. Yau and Dazhi Huang

(School of Computing, Informatics and Decision Systems Engineering, and
Information Assurance Center, Arizona State University, Tempe, AZ, USA)

Abstract Computing paradigms have evolved towards integrating small and personal computing devices with large-scale and resource-sharing computing infrastructures, such as services and cloud computing (SCC) platforms during the past two decades to provide users anytime, anywhere computing capability with more flexibility, richer computing resources, higher cost-effectiveness, lower risks of system failures, and less management and maintenance effort. Situation-aware (SA) applications in SCC environments have the capability to adapt SCC systems to situation changes to facilitate better human-computer interactions, continuous system availability, and satisfactory quality-of-service (QoS) for the applications. Mission-critical applications in many domains usually require situation awareness, and have started to incorporate it in SCC platforms due to the need of rapidly reconfiguring and integrating systems owned and operated by various organizations for sharing information and coordinating operations. In this paper, the concepts and characteristics of SCC and situation awareness (*SAW*), and the challenges and current state of the art in developing SA applications in SCC environments are discussed. Our research in this area, including the development of Adaptable Situation-aware Secure Service-based (AS^3) systems and Adaptive Service-Based Systems (*ASBS)* with QoS Monitoring and Adaptation (M/A), is presented. Future research in this area is discussed.
Key words: situation-aware applications; services computing; cloud computing; QoS monitoring and adaptation; development and runtime support

Yau SS, Huang DZ. Development of situation-aware applications in services and cloud computing environments. *Int J Software Informatics*, **Vol.7, No.1 (2013): 21–39. http://www.ijsi.org/1673-7288/7/i147.htm**

1 Introduction

Computing paradigms have changed substantially during the past two decades, and have evolved towards integrating small and personal computing devices, such as desktop PCs, mobile and embedded devices, with large-scale and resource-sharing computing infrastructures, such as services and cloud computing (SCC) platforms. More and more workloads have shifted from small computing devices at the frontend to computing infrastructures at the backend for various demanding applications. Nowadays, people can conveniently manage their data and access a variety of software services online. Such changes in computing paradigms have provided users anytime, anywhere computing capability with more flexibility, richer

Corresponding author: Stephen S. Yau, Email: yau@asu.edu
Received 2012-10-01; Revised 2012-12-28; Accepted 2013-01-08.

computing resources, higher cost-effectiveness, lower risks of system failures, and less management and maintenance effort[1]. However, this trend in computing paradigms has also stimulated new applications and posed new challenges for software engineering in new computing environments[1].

In this paper, we will address the development of one important type of new applications, called *situation-aware*(SA)*applications,* in SCC environments. SA applications in SCC environments have the capability to adapt to situation changes to facilitate better human-computer interactions, continuous system availability, and satisfactory quality-of-service (QoS) for the applications. Situation changes may include changes in physical and operational contexts, such as weather, user locations, and available computing resources, and changes in more complex user-specified conditions based on historical and current contexts, which often reflect important status changes of SCC systems and user operations and require immediate response actions. Situation awareness (SAW) is very important for current and future computing systems, and particularly useful for users in dynamically changing environments, e.g. mobile users in SCC environments. Mission-critical applications in many domains usually require SAW, such as emergency management and response, environmental monitoring and control, healthcare, and homeland security. Many of these applications have adopted or started the transition to SCC platforms due to the need of rapidly reconfiguring and integrating systems owned and operated by various organizations for sharing information and coordinating operations. For example, an emergency management and response system needs to have the capabilities to continuously function after major disasters, such as earthquakes and hurricanes, and perform system adaptation in order to deliver essential information and services with satisfactory QoS to various organizations participating in disaster relief operations, which requires the system to be situation-aware and dynamically reconfigurable.

The development of SA applications in SCC environments poses many challenges, which include context management, situation analysis, generating situation-triggered response, and development and runtime support for SA applications in SCC environments. These challenges can be summarized and categorized as follows:

- ***Context discovery, acquisition and dissemination***

  C1) How to identify important contextual data relevant to improving the quality of end users' interactions with SCC environments?

  C2) How to effectively discover and gather relevant contextual data in dynamic and heterogeneous SCC environments without information overloading?

  C3) How to effectively share contextual data among various components of the applications to reduce system overhead and improve users' interactions without compromising security and privacy?

- ***Situation analysis and recognition***

  C4) How to model situations affecting end user experiences in SCC environments in a way that facilitates automated reasoning and decision making for application and system adaptation?

C5) How to efficiently analyze potentially large amount of raw contextual data distributed in SCC environments to recognize situations?

C6) How to exchange and share situation analysis results without breaching security and privacy?

– ***Situation-triggered response***

C7) How to identify appropriate actions in response to situation changes in a timely manner?

C8) How to identify and resolve (reconcile) conflicts among actions triggered by situation changes, especially when multiple SA applications are being used by various end users on shared resources?

– ***Development and runtime support for SA applications in SCC environments***

C9) How to analyze SA requirements and specify contexts, situations and associated actions in SCC environments?

C10) How to automate some development tasks, including code generation, testing, verification and validation, of SA applications in SCC environments?

C11)How to effectively support various runtime activities, including context management, situation analysis and action triggering, for SA applications in SCC environments?

The rest of this paper will be organized as follows: A brief overview of SCC and *SAW*, and the current state of the art in developing SA applications in SCC environments will be presented in Section 2. Our approach to the rapid development of Adaptable Situation-aware Secure Service-based (AS^3) systems will be presented in Section 3. Our approach to the development of Adaptive Service-Based Systems (*ASBS*) with QoS monitoring and adaptation (M/A) will be presented in Section 4. Future research directions in this area will be discussed in Section 5.

2 Background and Current State of the Art

There has been substantial research on services computing, cloud computing, and situation awareness. In this section, we will briefly discuss the concepts of services and cloud computing, context and situation awareness, and the current state of the art in developing SA applications.

2.1 Services and cloud computing (SCC)

Services and cloud computing have advanced rapidly over the past two decades to enable the rapid development and deployment of large-scale distributed applications in various domains, such as e-business, enterprise computing infrastructures, health information systems, collaborative research and development, and homeland security. Both services and cloud computing view computing resources as services, and enable resource sharing and outsourcing through service discovery, composition, and delivery. While services computing focuses on architectural design enabling application development through service discovery and

composition, cloud computing focuses on the effective delivery of services to users through resource virtualization and load balancing[1].

The key concept of services computing is service-oriented architecture *(SOA)*[2], an architectural model for creating, sharing, discovering, and accessing services, which are reusable and self-contained software entities with well-defined and interoperable interfaces to provide certain functionalities over networks following standard protocols. Three types of stakeholders are usually involved in the development of *SOA*-based applications: *service providers* developing and hosting services, *service consumers* using services in their applications, and *service brokers* facilitating the publishing, management and discovery of services. Standards like *WSDL*, *SOAP*, and *UDDI* used in *SOA* enable the description and exchange of service information among stakeholders, and the publishing and discovery of services by stakeholders. *SOA* has several unique characteristics that enable rapid development of large-scale distributed applications. First, services are loosely-coupled and stateless, *i.e.*, there are no direct dependencies among individual services and no state information specific to activities on services. Second, services can be developed and managed independently by organizations using various languages and platforms. Third, services can be discovered and composed by application developers following specific workflows to provide complex functionalities.

Cloud computing systems are considered as highly scalable distributed systems enabling on-demand ubiquitous access over networks for users to a shared pool of virtualized and configurable computing resources, such as servers, storage, and software services. They are usually used for large-scale distributed applications for many users. As the cloud computing paradigm is quickly adopted in various applications, users may soon be able to access computing capabilities and resources as needed over wired or wireless networks anytime, anywhere using various computing devices, such as tablets, smart phones, and desktop PCs.

As indicated in the recent NIST definition for cloud computing[3], there are five essential characteristics, three service models, and four deployment models for cloud computing. The five characteristics include *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity*, and *measured service*. The three service models include *Software as a Service (SaaS)*, *Platform as a Service (PaaS)*, and *Infrastructure as a Service (IaaS)*. The four deployment models include *private cloud*, *community cloud*, *public cloud*, and *hybrid cloud*.

2.2 Context and situation awareness

Context and situation are two closely related concepts. Although context is sometimes considered the same as situation[4,5], context and situation are normally considered different to facilitate the development of adaptive applications in highly dynamic environments[6-9]:

– *Context*: Various definitions for context have been made[6-10]. For examples, Dey, et al.[10] defined context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Chen, et al.[6]

defined context as "the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user". Based on various definitions of context, a context has the following properties: (1) A context changes over time and is meaningful only when it is associated with a time instant. (2) A context must be detectable with appropriate hardware or software support so that it could be used in computing. (3) A context must be relevant to the interaction between a user and an application. Based on these properties, Yau, et al.[8,9] defined context as "any instantaneous, detectable, and relevant property of the environment, the system, or users".

– *Situation*: In Refs. [8,9], a *situation* is defined as a set of context attributes of users, systems and environments over a period of time affecting future system behavior. Although there are other definitions for situation[11,12], it is commonly agreed that "situation" is a higher-level concept built upon "context", and a situation provides a higher-level understanding of the status of users or other objects involved in computing than a context.

The term "situation awareness" was first used in military as an important decision factor for pilots in 1970s according to a review of the factors determining the outcomes in air-to-air combats[13]. Since 1990s, situation awareness has been studied in many areas, including artificial intelligence, data fusion, and human-computer interactions. In the same period, researchers in mobile and ubiquitous computing introduced the concept of context-/situation-aware computing, which has become quite popular as a major feature of mobile and ubiquitous computing. The term "context-aware computing" was first introduced in 1994[14], and initially defined as the "ability of a mobile user's applications to discover and react to changes in the environment they are situated in". Dey, et al,[15] refined this definition and defined a context-aware computing system as a system using context to provide information and/or services relevant to the user's task. Similar to context-aware computing, situation-aware computing can be defined as the ability of being aware of situations and adapting the system's behavior based on situation changes[8,9]. Context-/Situation-aware computing is very useful in various mobile and ubiquitous computing applications, which are usually categorized based on how context and situation information is used in applications. Example categories include *proximate selection*, *automatic reconfiguration*, *contextual information and commands*, and *context/situation-triggered actions*[16].

The importance of context-/situation-aware computing lies on the new way of considering human-computer interactions in context-/situation-aware computing systems. Unlike traditional computing systems operating on explicit inputs (data and commands) from users, context-aware computing systems consider contexts as implicit inputs and operate on both contexts and the explicit inputs from users[17]. Hence, the interactions between human users and computers in context-aware computing are no longer only initiated by human users. Changes in contexts can also trigger and guide interactions between users and computers. This is a very useful feature for mobile users, where computing is expected to be invisible or distraction-free to users.

2.3 Developing SA applications in SCC environments

Earlier research on *SAW* focused on modeling and reasoning with *SAW*[8,18-23], and developing toolkits and frameworks for providing development and runtime support for SA applications in mobile and ubiquitous computing environments[8,9,15,24-29]. Many of these results can be adapted to SCC environments, especially those on context/situation modeling and situation analysis.

Recently, substantial research has been done for developing SA applications in services computing environments. Models for *SAW* in services computing environments, such as ContextUML[30] and SAW-OWL-S[31], have been developed to support the modeling and specification of context/situation awareness requirements in service-based systems (SBSs). ContextUML was developed as an integral part of service modeling to support the specification of context information as well as the binding between contexts and context-aware objects that provide context-aware actions[30]. SAW-OWL-S is an OWL-based service specification language, which incorporates *SAW* in service specifications[31] and provides semantic-rich expressions of context and situation information. Various approaches and frameworks have been developed to facilitate the development of context/situation-aware SBSs. Adaptable Situation-aware Secure Service-based (AS^3) systems support declarative specification and static analysis of *SAW* requirements and situation-aware access control policies, and automated composition and execution of situation-aware workflows[32-39]. CA-SOA (Context-Aware Service-Oriented Architecture)[40] uses ontologies to model contexts, and provides support for context-aware service discovery and access. In Ref. [41], a framework for context-aware adaptable web services, which uses SOAP message header for context dissemination, was presented. Besides various general models and techniques for *SAW* in service-based environments, much research has also been devoted to the development of domain-specific frameworks for context/situation-aware applications in service-oriented environments[42-44]. The ESCAPE framework[42] supports the sensing, storing, and sharing of contexts in web service based systems integrating mobile devices and computing infrastructures for disaster management. Mobile Location-aware Decision Support System[43] is an ontology-based service-oriented architecture for emergency management in mass gatherings. Omnipresent[44] is a web service based system providing context-aware location-based services. However, there still lacks a comprehensive approach to developing SA applications in SCC environments with runtime capability to ensure that the users' requirements on various QoS aspects, such as timeliness, accuracy, and security of SA applications, will be satisfied in dynamically changing environments. Techniques for monitoring and adapting service QoS in SCC environments need to be developed and incorporated in SA applications.

3 Rapid Development of AS^3 Systems

Our techniques for the rapid development of AS^3 systems includes a declarative model for *SAW* and the underlining AS^3 logic and calculus, the system architecture and software agent execution platform for AS^3 systems, the algorithms for automated situation-aware workflow composition and *SAW* agent synthesis, and a situation-

aware access control framework[32-39,45-48]. Our approach to developing AS^3 systems is a unified logic-based approach focusing on synthesizing software agents for situation-aware workflows and security policy enforcement with consideration of workflow QoS and resource allocation during the synthesis process.

An AS^3 system is a collection of services, users, processes and resources, which operates together to achieve users' goals under dynamic situations, including satisfying users' security policies[33]. Figure 1 shows the architecture of an AS^3 system, in which service providers publish their capabilities as services. Each service provides a set of methods as "actions" in the AS^3 system. *SAW* agents collect context data periodically, analyze situations based on context data and executed action results, trigger appropriate actions based on situation changes, and provide situational information to other agents for situation analysis, service coordination, and security policy enforcement. *Security* agents enforce relevant security policies in a distributed manner based on the current situation. *Mission Planning* and *Workflow Scheduling* services generate and schedule workflows to achieve users' goals based on security policies, situations and available resources. *Workflow* agents coordinate the execution of workflows based on situational information.

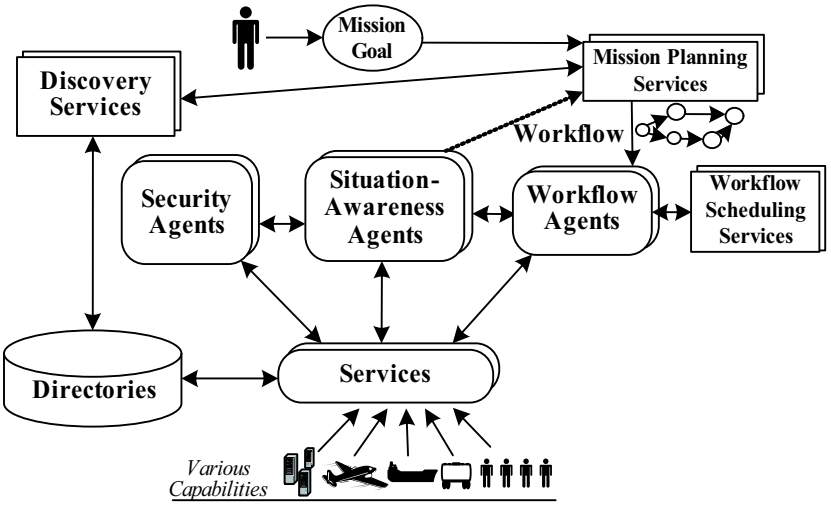

Figure 1. The architecture of an AS^3 system

3.1 Our approach to rapid development of AS^3 systems

Figure 2 shows our approach to rapid development of AS^3 systems[36-39,45,47,48], which consists of the following steps:

S1) Developers declaratively specify the user's mission goals, services and various requirements of the AS^3 system, including the specifications of *SAW*, security policies, timing and resource constraints, and failure handling, for the AS^3 system using AS^3 logic[36,48].

S2) Based on the *SAW*, service and security policy specifications, our AS^3 logic proof system automatically synthesizes *SAW* agents for collecting and processing situation information[37], and security agents for enforcing security policies in the

AS^3 system[45]. The synthesized *SAW* agents and security agents are described in AS^3 calculus terms[37,45].

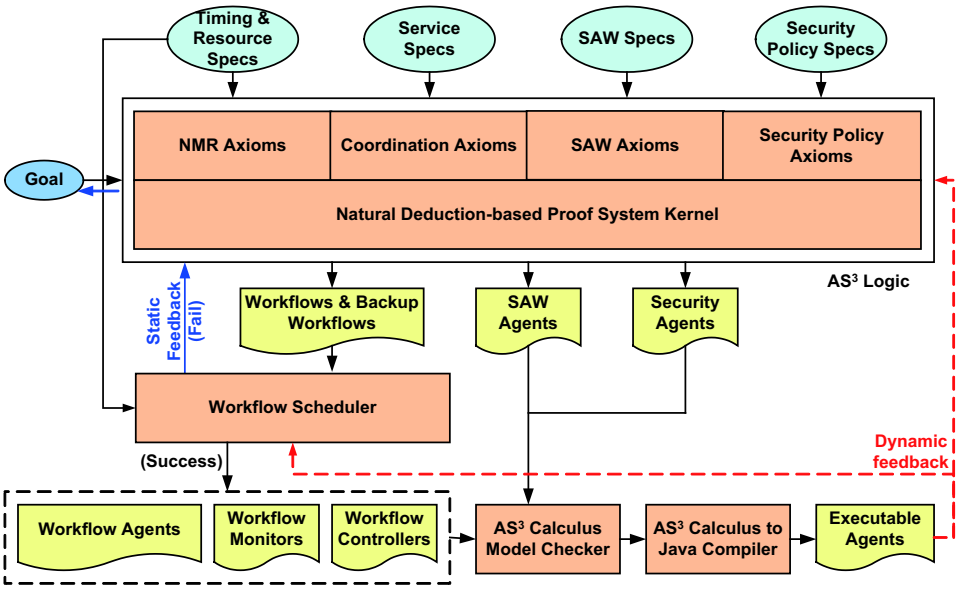

Figure 2. Our approach to rapid development of AS^3 systems

S3) Based on the specifications of the user's mission goals, available services and *SAW* requirements, our AS^3 logic proof system automatically synthesizes situation-aware workflows to achieve the mission goals[36]. Since runtime updates to certain Non-Monotonic Predicates (NMPs), such as changes in the cost or availability of services, may cause failures in execution of the synthesized workflows, a planning-phase mechanism is devised to accommodate possible failures learned from workflow execution history[38,39]. For each synthesized workflow, a set of backup workflows will be automatically synthesized by a risk management system integrated in our AS^3 logic proof system to reduce the risk of the failure of the main workflow. If the values of certain NMPs are different from their expected values in runtime, these backup workflows will be executed in parallel with the main workflow to compensate the impacts of the unexpected values of the NMPs.

S4) Based on the specified timing and resource constraints and security policies, our workflow scheduler schedules the workflows generated in ***S3)*** and automatically synthesizes workflow agents, monitors and controllers described in AS^3 calculus terms for monitoring, executing and controlling the workflows in the AS^3 system[47]. If the workflow scheduler successfully schedules the generated workflows and synthesizes the corresponding agents, go to ***S6)***; otherwise, go to ***S5).***

S5) If the workflow scheduler fails to schedule the generated workflows due to any violations of the specified timing and resource constraints or security policies, the workflow scheduler provides feedback to indicate the causes of the failure to the proof system for synthesizing alternative workflows. If any alternative workflows are synthesized, go back to ***S4)***; otherwise, the proof system notifies affected users that their mission goals cannot be achieved and need to be modified.

S6) Developers verify the agents generated in ***S2)*** and ***S4)*** using a model checker, and compile them to executable agents running on an agent platform.

AS^3 logic and calculus[36,48] provide the logical foundation of our approach. A user's service composition goal and the related *SAW* requirements are specified in an AS^3 logic formula describing a control flow graph and a set of situational constraints. Automated situation-aware service composition is achieved by constructing a proof for the user's service composition goal using the AS^3 logic proof system[36,39,48]. The constructed proof satisfies the given control flow and situational constraint specifications, and corresponds to a workflow which coordinates available services in the SBS to achieve the user's goal. *SAW* and workflow agents described in AS^3 calculus terms are synthesized from the constructed proof to execute the workflow. We have constructed a compiler for the AS^3 calculus terms to executable agents on a virtual machine-based agent platform, which was extended from the Secure Infrastructure for Networked Systems (SINS) platform[62].

During runtime, the information on the execution status of the workflows is collected by the workflow monitors. Based on such information and other situation information collected by *SAW* agents, the workflow controllers handle various failures occurred. If a failure occurred is one of the possible failures considered in ***S3)***, the workflow controllers load new workflow agents to run the backup workflows synthesized in ***S3)***. When severe failures which cannot be handled by the workflow controllers occur, the workflow controllers provide feedback to the proof system and the workflow scheduler to perform global re-planning and re-scheduling for the failed workflows.

3.2 Our declarative model for SAW in SBS

In this subsection, we will summarize our declarative model for *SAW*, called *SAW*model[34,37,48], which facilitates developers to analyze and specify situational constraints of service compositions in a SBS in a hierarchical and graph-based manner. Our *SAW* model is represented by an ontology as shown in Fig. 3. The ontology can be divided into three layers. The bottom layer *(L1)* shows the basic constructs for defining contexts and services, which will be further used to define *atomic situations*, the simplest form of situations. The middle layer *(L2)* shows the constructs for defining more complex situations. The top layer *(L3)* shows the relations between situations and service invocations, which will be used to specify system behaviors in response to situation changes. The constructs in the ontology are summarized as follows:

- A *context* has a unique *context name*, a *context type* and a *context value* at a time.
- A *context comparator* is a binary operator returning a Boolean value.
- A *service* has a unique *service name*, and is on a *host*.
- A *service invocation* is provided by a service, and has a unique *method name*, accepts inputs as arguments and returns outputs as context values.
- An *argument* can be a *constant* in the context value domain, or a *context variable* whose value is obtained through service invocations at runtime.

– An *atomic constraint* is used for comparing two arguments using a context comparator.

– A *situation* can be an *atomic situation*, a *logical composite situation* or a *temporal situation*. The value of a situation is a Boolean value.

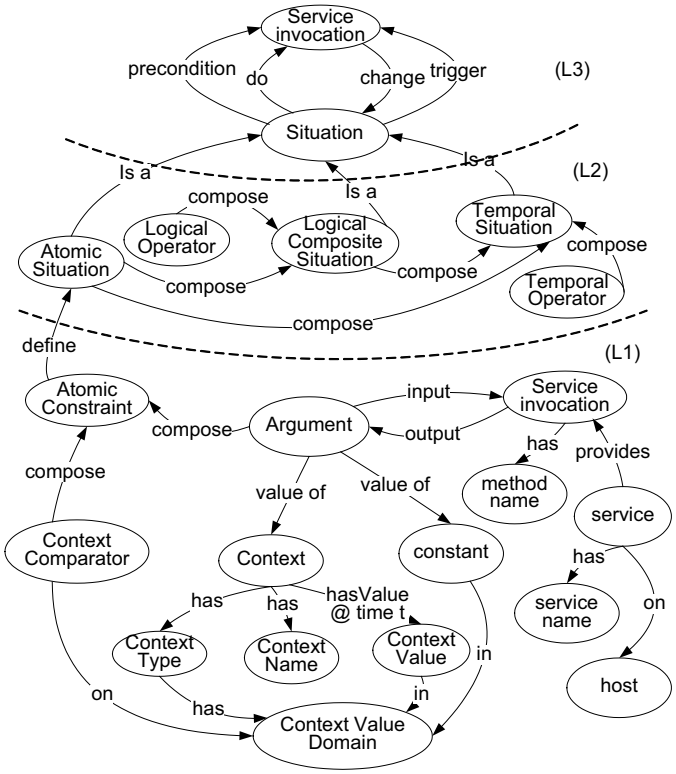

Figure 3. An ontology for our *SAW* model in SBSs

– An *atomic situation* is a situation defined using a set of service invocations and an atomic constraint, and cannot be decomposed into any other atomic situations.

– A *temporal operator* is either P, which means that the operand of P had been true over a period time in the past, or H, which means that the operand of H was true sometime in the past, defined over a period of time in the past.

– A *logical composite situation* is a situation recursively composed of atomic situations or other logical composite situations or temporal situations using logical operators, such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation).

– A *temporal situation* is a situation defined by applying a temporal operator on a situation over a period of time. The situation used to define a temporal situation can be either an atomic situation or a logical composite situation, which is not composed by any temporal situations.

Three basic relations, *precondition*, *do*, and *trigger*, are defined among situations and service invocations. Relation *precondition* describes a situation as a *precondition* of a service invocation. Relation *do* describes the effect of a service invocation. Relation *trigger* represents a reactive behavior of the system. In SBS, we assume that context data can be retrieved by invoking one or more services provided by the system platform or developed by various service providers.

To facilitate the specification of *SAW* requirements, graphical representations for the constructs in our *SAW* model and a GUI tool based on the graphical representations have been developed to facilitate developers to model *SAW* requirements visually. With our *SAW* model, developers do not need to learn or understand the complex syntax and semantics of AS^3 logic. Situational constraints for service composition are extracted from the graphical representations, and translated to the corresponding AS^3 logic specifications.

4 Development of *ASBS* with QoS M/A

Our techniques for the development of *ASBS* with QoS M/A includes a system modeling approach to constructing *Activity-State-QoS (ASQ)* models for QoS estimation, techniques for dynamic resource allocation and tradeoff handling between security and service performance, a system architecture for distributed QoS M/A and workflow execution, and *SOA*-compliant simulation techniques to support the validation of *ASBS* design[49-61]. Our approach to developing *ASBS* with QoS M/A focuses on modeling system dynamics related to service and workflow QoS, and providing runtime support for monitoring and controlling workflow QoS as well as adapting resource allocation, which is not provided in AS^3 systems.

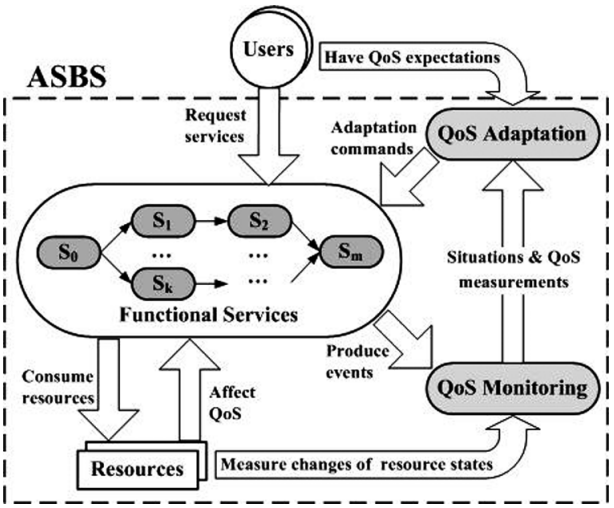

Figure 4. A conceptual view of *ASBS*

Figure 4 shows a conceptual view of our *ASBS*, in which functional services in the *ASBS* and the modules for QoS M/A form a closed control loop[53,54]. The QoS monitoring modules collect the measurements of various QoS aspects as well as relevant system status, which are used by the QoS adaptation modules to adapt

service compositions and/or service configurations in *ASBS* to satisfy various QoS requirements simultaneously.

Many QoS M/A activities in *ASBS* are based on Activity-State-QoS *(ASQ)* models, which reflect the cause-effect chain of user activities, system resource states, and QoS performance[49,53]. A service request from a user calls for a system process, which will utilize certain resources and change the states of the resources in the system environment. The changes of the resource state in turn affect the QoS of the process. We define an ASQ model as a 6-tuple

$$\langle A, S_0, S, Q, R_S, R_Q \rangle,$$

where A is a set of possible user activities which can be performed on a particular service, S_0 is a set of initial system resource states, S is the set of all possible system resource states, Q is the set of possible values for a particular aspect of QoS, R_S is a relation defined on $\mathrm{A} \times S_0 \rightarrow S$ representing how user activities and initial system resource states affect future system resource states, and R_Q is a relation defined on $\mathrm{A} \times S_0 \rightarrow Q$ representing how user activities and initial system resource states affect QoS aspects.

Our approach to developing *ASBS* consists of the following three major steps[49,53]:

S1) Gather the knowledge of the underlying cause-effect dynamics that drive the performance and tradeoffs among various QoS features, including timeliness, throughput, accuracy and security, based on controlled experiments and data analysis.

S2) Develop QoS M/A capabilities in *ASBS* based on the knowledge gathered in ***S1)***.

S3) Validate the QoS M/A capabilities developed in ***S2)*** through *SOA*-based simulations.

A system modeling approach[56,60] to constructing ASQ models for QoS estimation has been developed for ***S1)***, and applied to build accurate ASQ models for several example services[56,60]. For ***S2)***, techniques enabling system adaptation, including the techniques for dynamic resource allocation[52], and tradeoff handling between service performance and security[51,57] have been developed. A system architecture for distributed QoS M/A and workflow execution[58,59] along with a code generation tool for generating part of QoS M/A components has also been developed. For ***S3)***, modeling techniques and an *SOA*-compliant simulation environment have been developed to support the validation of *ASBS* design[50,55].

4.1 System architecture of ASBS

Figure 5 depicts the system architecture of our *ASBS*, which contains all the necessary modules to form a closed control loop for QoS M/A. The rectangles with solid lines are the basic software modules for QoS M/A in our *ASBS*. The rectangles with dashed lines are plug-ins for adapting service and workflow QoS to satisfy users' requirements[61]. The rounded rectangles are the functional services in an *ASBS*, which are from various providers. Each workflow in an *ASBS* has a workflow agent (*WA*) and a workflow-level QoS coordinator (*WQC*) for M/A activities at the workflow level. Each server in an *ASBS* has a server-level QoS coordinator (*SQC*)

and a server monitoring service (*SMS*) for M/A activities at the service and server levels. The basic software modules for QoS M/A are described as follows:

– *Workflow agent* (*WA*): A *WA* provides the mechanisms for controlling workflow execution and tracking workflow execution status. A *WA* encodes the control structure of a workflow and several specified patterns of possible structural changes for QoS adaptation.

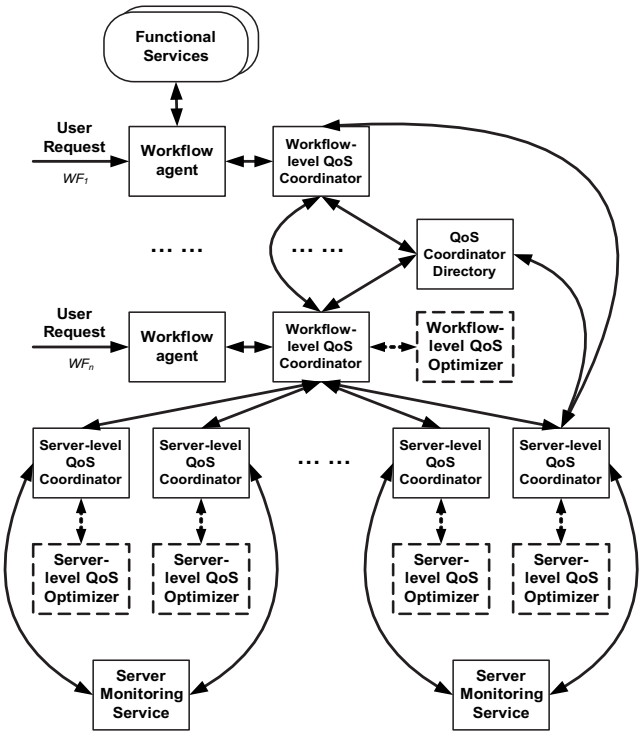

Figure 5. The system architecture of our *ASBS*

– *Workflow-level QoS Coordinator* (*WQC*): A *WQC* aggregates workflow-level QoS, detects situations requiring adaptation, and provides basic decision support for QoS adaptation, including finding alternative service instances and selecting one of the specified patterns of possible structural changes. A *WQC* also coordinates the activities performed by other M/A modules.

– *Server-level QoS Coordinator* (*SQC*): An *SQC* manages the information on services deployed on the server. An *SQC* measures QoS of the hosted services, and provides basic decision support for generating appropriate service configurations following a prioritized FCFS strategy.

– *QoS Coordinator Directory* (*QCD*): A *QCD* stores information of all workflow-/server-level coordinators in an *ASBS* to facilitate the dynamic discovery and coordination of *SQCs* and *WQCs*. The stored information includes communication endpoints, services managed by each *SQC*, and types of services to be used in a workflow handled by a *WQC*.

– *Server Monitoring Service* (*SMS*): An *SMS* monitors system resource status of a server, and provides remote access interface for retrieving such information.

A coordination protocol has been developed[59] for these modules and services in *ASBS* to execute workflows, and monitor and control workflow execution in ASBS.

4.2 A code generator for workflow agents and workflow-level QoS coordinators

Among the QoS M/A modules in *ASBS*, *WAs* and *WQCs* are workflow-specific. There will be one *WQC* and one *WA* for each workflow, and it is necessary to provide support to developers for rapid development of their workflow applications. Although various workflows often have significantly different control structures and QoS requirements, *WAs* and *WQCs* still share many common characteristics. Hence, we have constructed templates for *WAs* and *WQCs* using C#. These templates define the general structure of *WAs* and *WQCs*, and keep a lot of reusable code for *WAs* and *WQCs*. Tags are added in the templates to mark the places to be expanded when specific system information and user requirements, including service and workflow specifications, component deployment, and QoS of interests to users, are given. Based on these templates, we have developed a code generator for automatically generating codes for *WAs* and *WQCs*. This code generator can generate most part of *WAs* and *WQCs*, significantly reducing the programming effort of developers. The code generator is a console-based, stand-alone application that takes service, workflow and QoS specifications along with other system settings as inputs, and converts the general templates for *WAs* and *WQCs* to near-complete implementation of *WAs* and *WQCs* for specific workflows.

Figure 6 depicts the process incorporated in our code generator to generate the code for *WAs*and *WQCs*. This process consists of the following four main steps:

1. The code generator loads system information and user requirements specifying services, workflows and QoS of interest for each workflow from data files provide by the developers based on their system configuration and application requirements.

2. The code generator extracts information from the specifications to generate workflow and service tables.

3. The code generator loads templates for *WAs* and *WQCs*, including WQCDataTemplate, WQCGUITemplate and WFAgentTemplate, from a specified location where the templates are stored.

4. For each workflow in the workflow table:

   (a) Check if there is a QoS optimizer[62] available
   - If yes, check if the *WQC* to be generated for the current workflow is designated as the main coordinator, which will be responsible for interacting with the QoS optimizer.
     - If yes, go to *(iii)*
     - If no, go to *(ii)*
   - If no, go to *(ii)*

(b) Remove codes and annotations specific for the main coordinator from WQCGUITemplate and WQCDataTemplate, including data structures specific for the main coordinator, and codes and annotations related to the interaction with the optimizer.

(c) Generate codes for each annotation in WQCDataTemplate and WQCGUITemplate, and replace the annotations with the generated codes.

(d) Output WQCData and WQCGUI files for the workflow being processed.

(e) Generate codes for each annotation in WFAgentTemplate and replace the annotations with the generated codes.

(f) Output WFAgent file for the workflow being processed.

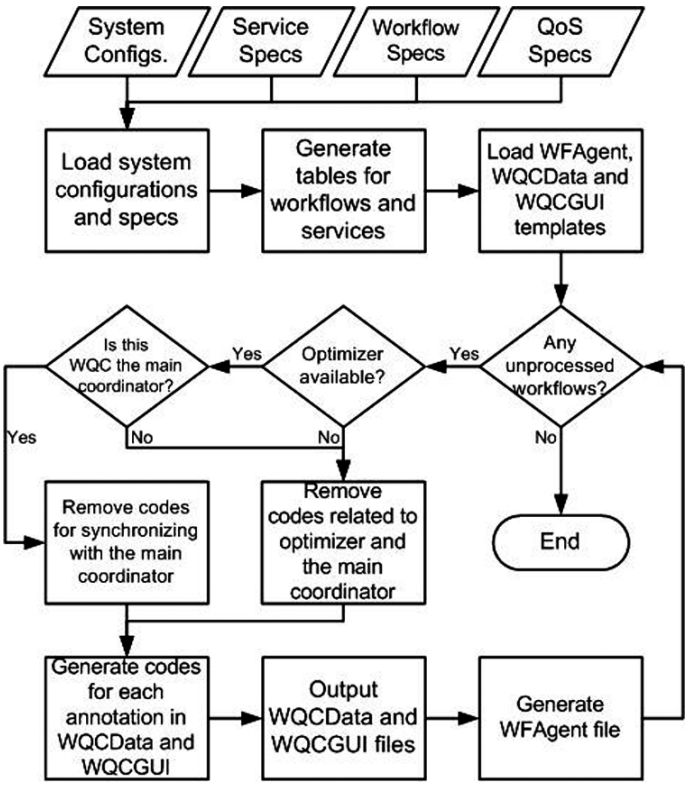

Figure 6. The process of our code generator for generating *WAs*and *WQCs*

After Steps *1) – 4)*, three files will be generated for each workflow: *[NAME]*WQCData.cs, *[NAME]*WQCGUI.cs, and *[NAME]*WorkflowAgent.cs, where *[NAME]* is the name of the workflow. The developers need to include these files with their C# projects and complete the codes following the instructions embedded in the generated files before generating executables for workflow agents and WQCs.

5 Conclusions and Future Research

In this paper, we have reviewed the concepts and characteristics of services and cloud computing and situation awareness, and discussed the challenges and the

current state of the art for developing SA applications in SCC environments. We have presented our two approaches important to developing SA applications in SCC environments to meet some of these challenges. One is for the rapid development of AS^3 systems, which provides a unified logic-based approach to automatically synthesizing software agents for *SAW*, security enforcement and workflow execution to achieve users' mission goals. The other is for the development of *ASBS* with QoS monitoring and adaptation, which provides a system architecture and techniques enabling distributed QoS M/A in service-based systems to ensure the satisfaction of users' QoS requirements in dynamically changing environments. Although the challenges ***C1***, ***C2***, ***C4***, ***C7***, ***C9***, ***C10*** and ***C11*** discussed in Section 1 have been addressed, many improvements need to be made and a number of important issues need to be resolved. Future research in this area may include the following aspects:

- Incorporation of fuzzy logic and semantic technology in *SAW* requirement specification and situation analysis to handle imprecise *SAW* requirements and uncertainty in users' behavior.
- Techniques for effective analysis and reconciliation of conflicting *SAW* requirements from multiple users on shared computing resources.
- Security protection and privacy preservation techniques for protecting sensitive context and situation information of users without serious impact on the usability of SA applications.
- Service composition techniques enabling more efficient adaptation of SA applications.
- Techniques for optimizing service performance and resource utilization for SA applications in SCC environments.
- Advanced middleware or frameworks providing strong development and runtime support for SA applications in SCC environments.
- Techniques for automating verification and testing of SA applications in SCC environments.
- Advanced simulation techniques to facilitate the design and rapid prototyping of SA applications in SCC environments.

Acknowledgment

The authors would like to thank Hasan Davulcu, Hessam Sarjoughian and Nong Ye of Arizona State University, Supratik Mukhopadhyay of Louisiana State University, and Ramesh Bharadwaj of US Naval Research Laboratory for their contributions to the research on AS^3 systems and *ASBS* with QoS M/A.

References

[1] Yau SS, An HG. Software engineering meets services and cloud computing. Computer, 2011, 44(10): 47–53.

[2] OASIS SOA Reference Model. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[3] The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-- 145/SP800--145.pdf

[4] Schiele B, Starner T, Rhodes B, Clarkson B, Pentland A. Situation aware computing with wearable computers. In: Barfield W, Caudell T, eds. Fundamentals of Wearable Computers and Augmented Reality. Lawrence Erlbaum Associates, Mahwah, NJ, 2001. 511.

[5] Mostefaoui GK, Pasquier-Rocha J, Brezillon P. Context-aware computing: A guide for the pervasive computing community. Proc. IEEE/ACS Int'l Conf. on Pervasive Services. 2004. 39–48.

[6] Chen G, Kotz D. A survey of context-aware mobile computing research[Technical Report, TR2000–381]. Dept. of Computer Science, Dartmouth College, Hanover, NH. 2000.

[7] Schmidt A. Ubiquitous computing: computing in context[Ph.D. Thesis]. Computing Dept., Lancaster University, UK. 2002.

[8] Yau SS, Wang Y, Karim F. Development of situation-aware application software for ubiquitous computing environments. Proc. of 26th IEEE Annual Int'l Computer Software and Applications Conf. 2002. 233–238.

[9] Yau SS, Karim F, Wang Y, Wang B, Gupta SKS. Reconfigurable context-Sensitive middleware for pervasive computing. IEEE Pervasive Computing, 2002, 1(3): 33–40.

[10] Dey AK, Abowd GD. Towards a better understanding of context and context-awareness [Technical Report, GIT-GVU-99–22]. Georgia Institute of Technology, Atlanta, GA. 1999.

[11] Marti P, Gabrielli F, Pucci F. Situated interactions in art. Personal and Ubiquitous Computing, 2001, 5(1): 71–74.

[12] Selker T, Burleson W. Context-aware design and interaction in computer systems. IBM System Journal, 2000, 39(3–4): 880–891.

[13] Watts BD. Clausewitzian friction and future war (revised edition). Institute of National Strategic Studies, National Defense University, Washington D.C., 2004. 57.

[14] Schilit BN, Theimer M. Disseminating active map information to mobile hosts. IEEE Network, 1994, 8(5): 22–32.

[15] Dey AK, Abowd GD. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 2001, 16(2–4): 97–166.

[16] Schilit BN, Adams N, Want R. Context-aware computing applications. Proc. of 1st IEEE Workshop on Mobile Computing Systems and Applications. 1994. 85–90.

[17] Pokraev S, Costa PD, Pereira Filho JG, Zuidweg M, Koolwaaij JW, van Setten M. Context-aware services: state of the art[Technical Report, TI/RS/2003/137]. Telematica Instituut, Netherlands. 2003.

[18] Matheus CJ, Kokar MM, Baclawski K. A core ontology for situation awareness. Proc. of 6th Int'l Conf. on Information Fusion. 2003. 545 –552.

[19] Matheus CJ, Kokar MM, Baclawski K, Letkowski J. Constructing RuleML-based domain theories on top of OWL ontologies. Proc. of 2nd Int'l Workshop on Rules and Rule Markup Languages for the Semantic Web. 2003. 81–94.

[20] Chen H, Finin T, Joshi A. An Ontology for Context-Aware Pervasive Computing Environments. Knowledge Engineering Review, 2003, 18(3): 197–207.

[21] Wang XH, Zhang DQ, Gu T, Pung HK. Ontology based context modeling and reasoning using OWL. Proc. of Context Modeling and Reasoning Workshop, in conjunction with 2nd IEEE Annual Conf. on Pervasive Computing and Communication. 2004. 18–22.

[22] Yau SS, Wang Y, Huang D, In H. A middleware situation-aware contract specification language for ubiquitous computing. Proc. of 9th IEEE Int'l Workshop on Future Trends of Distributed Computing Systems. 2003. 93–99.

[23] Yau SS, Huang D, Gong H, Seth S. Development and runtime support for situation-aware application software in ubiquitous computing environments. Proc. of 28th IEEE Annual Int'l Computer Software and Applications Conf. 2004. 452–457.

[24] Roman M, Hess C, Cerqueira R, Ranganathan A, Campbell RH, Nahrstedt K. A middleware infrastructure for active spaces. IEEE Pervasive Computing, 2002, 1(4): 74–83.

[25] Chan ATS, Chuang SN. MobiPADS: a reflective middleware for context-aware computing. IEEE Trans. on Software Engineering, 2003, 29(12): 1072–1085.

[26] Yau SS, Karim F. An energy-efficient object discovery protocol for context-sensitive middleware for ubiquitous computing. IEEE Trans. on Parallel and Distributed Systems, 2003, 14(11): 1074–1084.

[27] Yau SS, Karim F. An adaptive middleware for context-sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments. Real-Time Systems, 2004, 26(1): 29–61.

[28] Yau SS, Karim F. A context-sensitive middleware-based approach to dynamically integrating mobile devices into computational infrastructures. Journal of Parallel and Distributed Computing, 2004, 64(2): 301–317.

[29] Yau SS, Huang D, Gong H, Yao Y. Support for situation-awareness in trustworthy ubiquitous computing application software. Journal of Software Practice and Engineering, 2006, 36(9): 893–921.

[30] Sheng QZ, Benatallah B. ContextUML: a UML-based modeling language for model-driven development of context-aware Web services development. Proc. of Int'l Conf. on Mobile Business, 2005. 206–212.

[31] Yau SS, Liu J. Incorporating situation awareness in service specifications. Proc. of 9th IEEE Int'l Symp. on Object and Component-Oriented Real-Time Distributed Computing. 2006. 287–294.

[32] Yau SS, Davulcu H, Mukhopadhyay S, Huang D, Yao Y. Adaptable situation-aware secure service-based systems. Proc. of 8th IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing. 2005. 308–315.

[33] Yau SS, Yao Y, Banga V. Situation-Aware Access Control for Service-Oriented Autonomous Decentralized Systems. Proc. of 7th Int'l Symp. on Autonomous Decentralized Systems. 2005. 17–24.

[34] Yau SS, Gong H, Huang D, Gao W, Zhu L. Automated agent synthesis for situation awareness in service-based systems. Proc. of 30th IEEE Annual Int'l Computer Software and Applications Conf. 2006. 503–512.

[35] Yau SS, Yao Y, Yan M. Development and runtime support for situation-aware security in autonomic computing. Proc. of 3rd Int'l Conf. on Autonomic and Trusted Computing. 2006. 173–182.

[36] Yau SS, Davulcu H, Mukhopadhyay S, Gong H, Huang D, Singh P, Gelgi F. Automated situation-aware service composition in service-oriented computing. Int'l Journal on Web Services Research, 2007, 4(4): 59–82.

[37] Yau SS, Gong H, Huang D, Gao W, Zhu L. Specification, decomposition and agent synthesis for situation-aware service-based systems. Journal of Systems and Software, 2008, 81(10): 1663–1680.

[38] Singh P, Gelgi F, Davulcu H, Yau SS, Mukhopadhyay S. A risk reduction framework for dynamic workflows. Proc. of 2008 IEEE Int'l Conf. on Services Computing. 2008. 381–388.

[39] Davulcu H, Mukhopadhyay S, Singh P, Yau SS. Default α-logic for modeling customizable failure semantics in workflow systems using dynamic reconfiguration constraints. Proc. of 2009 Int'l Conf. on Grid and Distributed Computing. 2009. 49–56.

[40] Chen IY, Yang SJ, Zhang J. Ubiquitous provision of context aware web services. Proc. of 2006 IEEE Int'l Conf. on Services Computing. 2006. 60–68.

[41] Keidl M, Kemper A. A framework for context-aware adaptable web services. Proc. of 9th Int'l Conf. on Extending Database Technology. 2004. 826–829.

[42] Truong HL, Juszczyk L, Manzoor A, Dustdar S. ESCAPE: an adaptive framework for managing and providing context information in emergency situations. Proc. of 2nd European Conf. on Smart Sensing and Context. 2007. 207–222.

[43] Haghighi PD, Burstein F, Al Taiar H, Arbon P, Krishnaswamy S. Ontology-based service-oriented architecture for emergency management in mass gatherings. Proc. of 3rd IEEE Int'l Conf. on Service-Oriented Computing and Applications (SOCA). 2010. 1–7.

[44] De Almeida DR, De Souza Baptista C, Da Silva ER, Campelo CE, De Figueiredo HF, Lacerda YA. A context-aware system based on service-oriented architecture. Proc. of 20th Int'l Conf.

on Advanced Information Networking and Applications. 2006. 205–210.

[45] Yau SS, Yao Y, Chen Z, Zhu L. An adaptable security framework for service-based systems. Proc. of 10th IEEE Int'l Workshop on Object-Oriented Real-Time Dependable Systems. 2005. 28–35.

[46] Yau SS, Zhu L, Huang D, Gong H. An approach to automated agent deployment in service-based systems. Proc. of 10th IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing. 2007. 257–264.

[47] Yau SS, Huang D, Zhu L. An approach to adaptive distributed execution monitoring for workflows in service-based systems. Proc. of 4th IEEE Int'l Workshop on Software Cybernetics, in conjunction with 31st IEEE Annual Int'l Computer Software and Applications Conf. 2007. 211–216.

[48] Yau SS, Mukhopadhyay S, Davulcu H, Huang D, Bharadwaj R, Shenai K. Rapid development of adaptable situation-aware service-based systems. In: Zhang LJ, ed. Web Services Research for Emerging Applications: Discoveries and Trends. IGI Global. 2010. 104.

[49] Yau SS, Ye N, Sarjoughian HS, Huang D. Developing service-based software systems with QoS monitoring and adaptation. Proc. of 12th IEEE Int'l Workshop on Future Trends on Distributed Computing Systems. 2008. 74–80.

[50] Sarjoughian HS, Kim S, Ramaswamy M, Yau SS. An SOA-DEVS modeling framework for service-oriented software system simulation. Proc. of 2008 Winter Simulation Conf. 2008. 845–853.

[51] Yau SS, Yin Y, An HG. An adaptive model for tradeoff between service performance and security in service-based environments. Proc. of 7th Int'l Conf. on Web Services. 2009. 287–294.

[52] Yau SS, An HG. Adaptive resource allocation for service-based systems. Int'l Jour. Software Informatics, 2009, 3(4): 483–499.

[53] Yau SS, Ye N, Sarjoughian HS, Huang D, Roontiva A, Baydogan M, Muqsith MA. Toward development of adaptive service-based software systems. IEEE Trans. on Services Computing, 2009, 2(3): 247–260.

[54] Jiang CH, Hu H, Cai KY, Huang D, Yau SS. An intelligent control architecture for adaptive service-based software systems. Int'l Jour. Software Engineering and Knowledge Engineering, 2009, 19(5): 653–678.

[55] Muqsith MA, Sarjoughian HS, Huang D, Yau SS. Simulating adaptive service oriented software systems. Simulation, 2011, 87(11): 915–931.

[56] Ye N, Yau SS, Huang D, Baydogan M, Aranda BM, Roontiva A, Hurley P. Models of dynamic relations among service activities, system state and service quality on computer and network systems. Information, Knowledge, Systems Management, 2010, 9(2): 99–116.

[57] Yau SS, Yin Y, An HG. An adaptive approach to optimizing tradeoff between service performance and security in service-based systems. Int'l Jour. Web Services Research, 2011, 8(2): 74–91.

[58] Yau SS, Huang D. Proactive monitoring and control of workflow execution in adaptive service-based systems. In: Wong WE, Cukic B, eds. Adaptive Control Approach For Software Quality Improvement. World Scientific Publishing, Singapore. 2011. 239.

[59] Yau SS, Huang D. Distributed monitoring and adaptation of multiple QoS in service-based systems. Proc. of 8th IEEE Int'l Workshop on Software Cybernetics, in conjunction with 35th Annual Int'l Computer Software and Applications Conf. 2011. 31–36.

[60] Ye N, Aranda BM, Yau SS, Huang D. Analysis and modeling of service impacts on system activities, resource workloads and service performance on computer and network systems. Information, Knowledge, Systems Management, 2012, 11(3): 255–274.

[61] Ye N, Yang S, Aranda BM. Semi-distributed optimization protocol in service-based systems. Proc. of 2011 US-Korea Conf. 2011.

[62] Bharadwaj R. Secure middleware for situation-aware naval C2 and combat systems. Proc. of 9th Int'l Workshop on Future Trends of Distributed Computing System. 2003. 233–240.