# A Software Cybernetics Approach to Deploying and Scheduling Workflows in Service-based Systems

[1]Stephen S. Yau, [1]Dazhi Huang, [1]Luping Zhu and [2]Kai-Yuan Cai

*[1]Arizona State University, Tempe, AZ 85287-8809, USA*
*{yau, dazhi.huang, luping.zhu}@asu.edu*
*[2]Beijing University of Aeronautics and Astronautics, Beijing 100083, China*
*kycai@buaa.edu.cn*

## Abstract

*Service-based Systems (SBS) are being adopted by many distributed systems. Applications in SBS can often be viewed as the composition of various computing services following specific workflows. These workflows often need to satisfy various timing and resource constraints. In this paper, a software cybernetics approach to deploying and scheduling workflows with timing and resource constraints in SBS is presented. In our approach, a logic-based technique for modeling and solving timing and resource constraints for workflows in SBS is developed to generate the initial resource assignments, schedules and deployment plans of agents for workflows. The principles and concepts in software cybernetics are applied to guide the synthesis of software controllers for monitoring and adapting system behavior.*

***Keywords:*** Software cybernetics, deployment, scheduling, workflows, service-based systems, and controller synthesis.

## 1. Introduction

Recently, the rapid development of service-oriented computing and grid computing have led rapid adoption of Service-Oriented Architecture (SOA) in distributed computing systems, such as grid-enabled applications, enterprise computing infrastructures, and Global Information Grid (GIG). The capabilities for rapidly composing distributed applications and enabling the integration of the "system of systems" are most favorable offered by SOA-based techniques over other techniques for distributed computing. Systems based on SOA, called *service-based systems (SBS)*, comprise of various computing services offered by providers over networks. Applications in such systems can be viewed as the composition of these services following specific workflows, which are sequences of coordinated and cooperated invocations of services in the systems to achieve users' goals. Techniques based on SOA to enable utility computing have emerged, and they become a cost-effective way for organizations to outsource their computing tasks to infrastructure providers and receive computing services on-demand.

The above developments related to SOA have brought forth many important research issues. Among these, the following issues are of particular interests for rapid development, optimized deployment and easy management of service-based systems:

*R1)* How to automate the service composition in SBS based on declarative specifications of users' requirements and system environments?

*R2)* How to automatically select proper resources for the distributed execution-components, namely agents, of workflows, deploy the agents on proper hosts distributed in networks, and schedule the execution of these agents to provide timely responses and optimize system performance?

*R3)* How to detect or predict violations of timing and resource constraints for certain workflows in runtime, and adapt the systems to overcome such violations?

In this paper, we will present a software cybernetics approach to deploying and scheduling workflows with timing and resource constraints in SBS. In our approach, a logic-based technique for modeling and solving timing and resource constraints for workflows in SBS is developed to perform off-line resource planning, which generates the initial resource

assignments, schedules and deployment plans of agents for workflows before executing the workflows. The principles and concepts in software cybernetics are applied to guide the synthesis of "controllers", which are software modules to monitor system behavior, and take proper control actions, if necessary, to ensure that timing and resource constraints will not be violated.

## 2. Current State of the Art

For the deployment of workflows in SBS, some related work has been done in component deployment and dynamic agent allocation. The Object Management Group (OMG) defined deployment as the processes between acquisition of software and execution of software, and presented a five-phase component deployment process [1]. In [2], a set of protocols was introduced for users to dynamically deploy service components on remote sites and reconfigure the deployment scheme through reflective stubs named Ambassadors. In [3], two dynamic agent allocation mechanisms for large-scale multi-agent systems were presented. One mechanism minimizes agent communication cost by reallocating some agents to another agent platform when communication patterns among agents change. The other mechanism prevents overloaded nodes from affecting overall system performance by migrating agents on overloaded nodes to other nodes. In [4], an active distributed monitoring system based on mobile agents was presented. During the execution of a monitoring task, the monitoring agents sense the environment and take actions to maintain location optimality based on a distributed algorithm for computing agent locations. However, these techniques are not designed for workflows in SBS, and hence do not take into consideration the temporal and situational dependencies among services to be invoked in workflows, which may also affect the deployment of workflows.

For workflows with timing and resource constraints, the workflows need to be properly scheduled with appropriate resources assigned to them. Classical scheduling algorithms, such as Rate Monotonic and Earliest Deadline First, depend on *a priori* knowledge of workload and systems, and hence they can only be used in predictable environments [5-6]. Dynamic scheduling systems, such as Spring [7], provide performance guarantees upon new task arrivals with on-line admission control, but require *a priori* task set characterizations. Feedback control real-time scheduling techniques can provide robust performance guarantees in unpredictable environments, but also require *a priori* task set characterizations and are not designed for workflow scheduling [8-11]. Logic-based workflow scheduling techniques can be used to handle multiple constraints on workflow execution, but cannot deal with unpredictable environments [12-13]. Grid workflow scheduling techniques can provide performance guarantees for workflows in Grid, but are designed for scientific computing (computation intensive, high parallelism, little coordination) [14-17].

## 3. Background

Our approach to deploying and scheduling workflows in SBS is partly based on our Adaptable Situation-aware Secure Service-based ($AS^3$) Systems, including a declarative model for situation awareness (SAW) [18-20], an $AS^3$ calculus and logic [19-20] for $AS^3$ systems. For sake of completeness, we will briefly review our declarative SAW model, and $AS^3$ calculus and logic.

Our declarative SAW model provides formal definitions of the essential constructs like contexts, situations, and relations among situations and actions [18, 20]. Graphical representations for the constructs in our SAW model have been introduced and implemented in a supporting GUI tool to enable rapid modeling of SAW requirements. Our SAW model is language-independent and can be translated to specifications of various formal languages, such as Transaction F-Logic and $AS^3$ logic.

$AS^3$ calculus [19] is based on classical process calculus. $AS^3$ calculus can model timeouts, failures, service invocations, and communications. A process can be an inactive process, parallel composition of two processes, a nominal identifying a process, or a process performing an external action or an internal computation. External actions involve input and output actions (as in the ambient calculus [21]) on named channels and migration of processes to another ambient. Internal computation involves beta reduction, conditional evaluation for logic control, and service invocations.

$AS^3$ logic has both temporal and spatial modalities for expressing situation information, as well as modalities for expressing communication and service invocation. It provides atomic formulas for expressing relations among variables and nominals for identifying agents. $AS^3$ logic allows declarative specification of QoS requirements, such as security, SAW, and real-time requirements. Models for the logic are processes in the $AS^3$ calculus. In addition, we can use primitive connectives and modalities in $AS^3$ calculus to define some useful connectives and modalities as follows:

- Eventually: $\mathrm{diam}(\varphi) := \mathrm{E}(\mathrm{T}\ U\ \varphi)$, where $\mathrm{E}(\mathrm{T}\ U\ \varphi)$ is a modality *until* provided in $AS^3$ logic

- Universal quantification on time: $\forall t\ \varphi := \neg\exists t \neg\varphi$

## 4. Problem Statement

To state the problem more clearly, we need to first discuss how workflows are executed in SBS. Figure 1 depicts the entities and their relations in SBS, in which all the ellipses represent the entities in the system, including *agent*, *service*, *host*, *domain*, *user*, *service provider*, and *domain administrator*. Each arrow represents a relation between two entities.

An *agent* is a software component owned by a *user* and acting on behalf of the *user* to use various *services*. A *service* is the capability provided by a *service provider* and provisioned by a *host*. A *host* is a computer in the network, where agents and services can be deployed, and belongs to a specific domain. A *domain* is a group of hosts administrated as a unit with common rules. Domains and hosts are managed by *domain administrators*. Each domain usually corresponds to a different organization with its own policies for resource usage.

The goal of a workflow consists of two parts. One specifies the functional requirements of the workflow, and the other specifies the non-functional requirements. There are many possible non-functional requirements. In this paper, we focus only on the requirements on timing and resource allocation.

An important aspect distinguishes our research with others is that we consider the entire system to be situation-aware. A workflow may have different requirements in different situations, which cause agents of this workflow to take different actions. Service providers may provide services with different behavior, require different resources, or provide different levels of QoS when situations change. Each domain or host may have different policies regulating the access to services on a certain host or within a certain domain in different situations. Such situation awareness allows more flexibility in SBS, and is very important for practical systems. However, most existing research in this area does not take situation awareness into consideration.

For SBS discussed above, we define the following:

- *Deployment plan*: A *deployment plan* of a workflow is the mapping between the agents for the workflow and the available hosts in SBS. Specifically, we consider a dynamic mapping, in which agents may migrate to different hosts when situations change.
- *Resource*: A *resource* in SBS is an object required by the invocation of certain services. A resource can be CPU time, memory, bandwidth, datasets or other objects related to computing and communication. In particular, services are also considered a special kind of resources since the invocation of a service may require the usage of other services in SBS.
- *Resource assignment*: A *resource assignment* of a workflow is the mapping between each service invocation in the workflow and the available resources that can be used by the service invocation. The resource assignment information will be used in runtime to manage the allocation or revocation of resources to/from workflows.
- *Schedule*: A *schedule* of a workflow is a complete description of the earliest start time and latest completion time of each service invocation in the workflow, given a particular resource assignment. A schedule of a workflow may change when the corresponding resource assignment for the workflow changes due to different characteristics of resources.
- *Controller*: A *controller* is an agent that monitors the system status and the behavior of other agents for workflows, allocates or revokes resources to/from agents, and controls the migration of agents among distributed hosts according to the timing and resource requirements. Controllers are synthesized automatically in our approach.

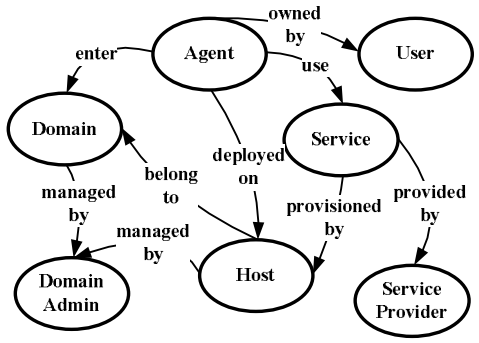


**Figure 1.** An ontology for various entities in SBS

With the above definitions, the problem we address in this paper can be rephrased as follows: Given the goal of a workflow, and the knowledge of a SBS, including the specifications of domains, hosts, services, and agents, we develop an approach to generating the schedule, resource assignment and deployment plan for the workflow, and synthesizing controllers to ensure that the goal of the workflow is met.

## 5. Overview of Our Approach

We have identified three important research issues R1)-R3) for rapid development, efficient deployment and easy management of SBS. For R1), the techniques based on AI planning and semantic web have been

developed and are being continuously improved to automate the service composition in SBS [22-24]. In this paper, we assume that the service compositions are either specified by developers or generated by some composition services, and focus our discussions on R2) and R3). For R2), substantial research has been done in operations research and workflow community. However, their results cannot effectively accommodate the highly dynamic nature of SBS, where services are loosely-coupled and operated by different organizations. In such environments, services may become unavailable without any notification to users, or may degrade the performance due to system overload or changed organization policies. Hence, using these services may cause violations of timing and resource constraints of the workflows. This leads to R3), which requires the capability of monitoring system behavior in runtime and adapting systems dynamically based on the collected feedback.

Our investigations on R2) and R3) show that these two closely-related issues can be addressed from a software cybernetics [25-26] perspective. Software cybernetics is an emerging research area concerning the interplay of software and control [26]. In software cybernetics, basic principles of control theory are applied to control the development process as well as the execution of software systems. R2) and R3) are essentially concerning the control of what and how resources are used in workflows, and when and how workflows and the underlying systems should be adapted in response to changes in execution environments so that the timing and resource constraints of the workflows will not be violated. Hence, the principles and concepts in software cybernetics are well-suited for dealing with these problems. We can model the agents, which execute workflows in SBS, as controlled objects, and model the adaptation of workflows or systems as the control applied on the controlled objects [26]. For deploying and scheduling workflows with timing and resource constraints in SBS, the goal of control is to avoid constraint violations and optimize system performance. However, agents for workflows are often executed concurrently. This makes it difficult to apply traditional control theories, which are mainly based on transfer functions and state-based models [26].

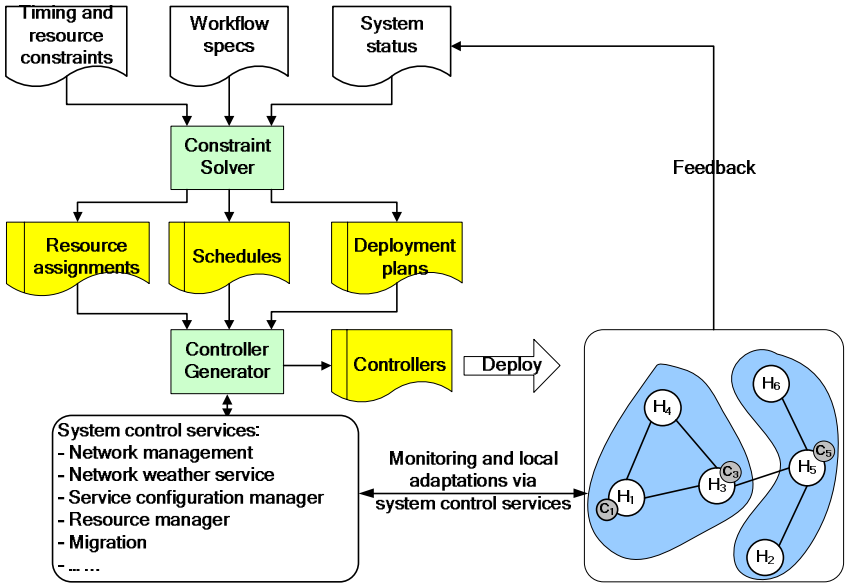


**Figure 2.** Our overall approach.

In this section, we will present an overview of our approach. Our approach consists of two interacting processes: the off-line resource planning, and online monitoring and adaptations, as shown in Figure 2.

The off-line resource planning process can be summarized as follows:

*P1)* The system status, and the goals of all the workflows in the system, including the timing and resource constraints, and workflow specifications, are taken as inputs of the constraint solver to generate proper resource assignments, schedules and deployment plans for the workflows.

*P2)* A controller generator takes the resource assignments, schedules and deployment plans generated in *P1)*, and synthesizes a set of controllers based on the available system control services. The suitable locations of the synthesized controllers are also determined in this process.

The generated controllers are then be deployed to the proper hosts in the target system for online monitoring and adaptation. The selection of hosts for deploying the controllers follows a simple principle: A controller is deployed on a host where the required feedback for the controller to determine proper control actions is collected. These controllers encode various constraints to be satisfied by the workflows as conditional evaluations in the controllers' internal logic. In runtime, the controllers use various *system control services* to monitor system status, determine whether there are constraint violations during the workflow executions, check whether there are opportunities to achieve better system performance, and invoke proper system control services to adapt the

workflow and system behavior. The adaptation performed by an individual controller is considered local adaptation since the controller only has knowledge of the host deploying the controller and the host's neighbors.

The controllers also continuously provide feedback to the offline resource planning process by updating the system status. In case any constraint violation cannot be resolved through local adaptations by the controllers, the controllers will provide the status information as feedback to the off-line resource planning process to perform a system-wise re-planning.

## 6. Solving Timing and Resource Constraints

In this section, we will discuss issues in solving timing and resource constraints in the off-line resource planning process.

- **Inputs for off-line resource planning**

As shown in Figure 2, the offline resource planning needs the following inputs to generate appropriate resource assignments, schedules and deployment plans, and the controllers are synthesized based on the generated results for adapting the system to satisfy users' requiremenets:

(a) *Timing and resource constraints,* which correspond to the requirements on expected response time from services, deadline of workflows, resources required for workflow executions, and situation-based access policy of resources.

(b) *Workflow specifications,* which provide information on the data and control flows in workflows.

(c) *System status,* which includes the descriptions of available services in the system, current situation, network traffic and system workload.

(d) *Information on system control services,* which describes the available system control services, including their interfaces, preconditions for using the services, and their effects in various situations.

In our approach, $AS^3$ logic is used to specify such inputs because $AS^3$ logic provides modalities for service invocations and communications among agents, and can support declarative specifications of SAW. In particular, (a) and (c) are specified using atomic formulas and atomic constraints in $AS^3$ logic, (b) is specified using *until*, *since*, *k* and *serv* in $AS^3$ logic [19], and (d) is specified using $AS^3$ logic the same way as other services used by workflows [19-20].

The models of $AS^3$ logic are processes in $AS^3$ calculus, and hence provide constructive interpretations for the logic, which is crucial to the synthesis of controllers. Such a logic-based approach to specifying the required inputs to the off-line resource planning process allows formal analysis on these inputs, such as verifying their consistency, and performing deductions on the inputs to infer new knowledge on resource allocations and real-time requirements.

- **Solving timing and resource constraints for workflows**

Once all the necessary knowledge are specified using $AS^3$ logic, deductions on the timing and resource constraints are performed based on system status and the data and control flows in workflow specifications. The deductions decompose the overall timing and resource requirements for a workflow to more specific requirements on each individual service invocation involved in the workflow, including earliest and latest time as well as the conditions for invoking a service.

The deduction is done based on a set of deduction rules similar to those presented in [12], except the following improvement for handling SAW:

In our approach, possible situation changes are examined in each step of the deduction to determine the proper constraints to be generated in the step. As discussed before, we consider the whole system to be situation-aware, and incorporate situational constraints of workflows and situation-dependent service behavior. This makes the decomposition of the overall timing and resource requirements more difficult because the workflows may take different execution paths and services may behave differently when situation changes. Our solution is to first convert the data and control flows in workflow specifications to a tree-like structure, and evaluate the possible situations in each node on the tree based on SAW and service specifications. When the workflow takes a certain execution path, the possible effects of the actions along the execution path are estimated based on service specifications, and used to update possible situation changes along the path. Hence, at a certain state of the execution of a workflow, we can generate the possible situations at the state based on the execution path(s) from the initial state to the current state. These situations will then be used in the deduction to determine the possible service behavior and the proper constraints to be enforced.

We also combine the above deduction process with the filtering of usable resources by eliminating resources not accessible or available due to certain situation-based access policies specified by the owners of the resources, or due to violations of constraints generated in a deduction step.

After the deduction process completes, the following results are generated:

- A set of candidate resources.

- The constraints on the resources to be used by each service invocation in the workflow.
- The earliest start time and latest completion time of each service invocation.

To solve this set of constraints over the candidate resources for the given workflow, many available constraint solving techniques can be used [27]. In our prototype implementation of the constraint solver shown in Figure 2, we have used the Constraint Handling Rule provided in XSB prolog [28] to develop the constraint solver. The generated resource assignments, schedules and deployment plans for a workflow are associated with various situations identified in the deduction process.

## 7. Synthesis of Controllers for Online Monitoring and Adaptations

The resource assignments, schedules and deployment plans for a workflow generated by the constraint solver are considered the *initial operations plan* for deploying and executing the workflow. Such an operations plan needs to be enforced properly in runtime and may be adapted if exceptional situations, such as system failure and overload, occur. As discussed in Section 5, this can be modeled as a control problem in the context of software cybernetics. In this section, we will present our preliminary ideas on the offline synthesis of a set of controllers to carry out and adapt the initial operations plan generated by our constraint solver described in Section 6.

Obviously, it is undesirable to use a centralized controller to perform all monitoring and adaptation tasks for workflows running in SBS, where network failures or unexpected long delay in communication may frequently occur. Proper distribution of these tasks to a set of controllers deployed on various hosts in the network will greatly improve the robustness of our system.

Currently, a simple distribution scheme is adopted in our approach: One controller for each host per workflow will be synthesized to perform the required monitoring and adaptations by invoking proper system control services when situation changes. These services are the system-specific facilities provided by underlying SBS for collecting system and network status information, and managing system configurations, network communications, computing resources, and agent migrations. Some examples of such system control services are Network Weather Service (NWS), APIs for managing thread priorities in operating systems, and migration services in agent middleware. The controllers for the same workflow coordinate their control activities by a simple token-passing protocol, which transfers the control among controllers by passing tokens from one controller to other controllers responsible for the successive actions in the workflow based on the progress of workflow execution. The tokens can be replicated by a controller to allow parallel execution of actions.

In order to synthesize such controllers, a Performance Index (PI) needs to be defined for assessing the system performance. We first define several useful functions. In the following definitions, we will refer the invocation of a service in a workflow as a task, and use $RA_i$ to represent the resource assignment for a task $i$. Since a task $i$ may require multiple resources, $RA_i$ is defined by a matrix $[RAV_0, \ldots, RAV_k]$, where $RAV_0, \ldots, RAV_k$ are called *resource assignment vectors (RAV)*. Each RAV has the form $[id, type, amount]^T$, where *id* is a unique identifier for the resource, *type* indicates the resource type, and *amount* is the minimum amount of resource *id* required by the task. In addition, $s$ stands for the current situation, $t$ stands for the current time, $h$ stands for the initial choice of the host where a task is performed, and $h'$ stands for the host where a task will be migrated to.

- *Readiness of task i*:

$r_i(s) = 1$ if the preconditions for task $i$ have been satisfied under situation $s$, and task $i$ is ready for execution.

$r_i(s) = 0$, if the preconditions for task $i$ have not been satisfied under situation $s$.

- *Single resource availability*:

$ra_m(s, t, RAV_m) = [1 - fail(id_m)] \times [1 - u(s, t, RAV_m)]$, where $RAV_m = [id_m, type_m, amount_m]^T$, $fail(id_m)$ is the probability that resource $id_m$ fails, and $u(s, t, RAV_m)$ is the probability that the available amount of $id_m$ is less than $amount_m$ at time $t$ under situation $s$. In our approach, $fail(id_m)$ is defined as the average time that resource $id_m$ is in failure state within a time unit. This can be obtained from system history data and is considered as a constant for resource $id_m$. $u(s, t, RAV_m)$ is estimated based on delays in the execution of other tasks requiring $id_m$, and the allocated amount of $id_m$ to those tasks.

- *Total resource availability for task i*:

$avail_i(s, t, RA_i) = \prod ra_m(s, t, RAV_m)$,
where $RA_i = [RAV_0, \ldots RAV_k]$, and $m = 0, \ldots, k$.

- *Gain from task i*:

$g_i(s, t, h, RA_i) = [t_{Li} - t - exec_i(s, h, RA_i)] \times avail_i(s, t, RA_i)$, where $t_{Li}$ is the latest completion time of task $i$, $exec_i(s, h, RA_i)$ is the estimated execution time of task $i$ on host $h$ under situation $s$ with resource assignment $RA_i$. Here, we assume that $exec_i(s, h, RA_i)$ has been obtained using system identification techniques [29]. A negative result from the *Gain* function indicates that the execution of task $i$ is delayed, and will lead to the violation of a timing constraint for the entire workflow.

Based on the above definitions, we define the PI for a task *i* as follows:

$PI_i(s, t, h) = r_i(s) \times [g_i(s, t, h', RA_i) - m_i(h, h')]$,

where $m_i(h, h')$ is the cost of migrating the task from host *h* to *h'*. In our approach, we define

$m_i(h, h') = \mu + size_i / bandwidth(h, h')$,

where $\mu$ is a constant representing non-communication cost for task migration, $size_i$ is the size of all information for task *i* that needs to be transferred from *h* to *h'* to complete the migration, and $bandwidth(h, h')$ is the available network bandwidth between *h* and *h'*.

Assume that tasks *0, ..., n* need to be performed on host *h* for a workflow *W*, the goal of the controller to be deployed on *h* for *W* is to maximize $\sum PI_i(s, t, h)$ at time *t* under situation *s* while keeping the result each individual *Gain* function non-negative, by determining appropriate $RA_i$ and migration target *h'* for each task.

To facilitate the synthesis of such controllers, the system control services are modeled using $AS^3$ logic in the same way as other services used by workflows [19-20], which allow the discovery of appropriate system control services through automated reasoning on the service specifications based on the tasks to be performed.

The synthesis process of a controller for a workflow is summarized as follows:

*1)* Extract necessary information for the functions used in the PIs of the tasks to be monitored by the controller from the initial operations plan of the workflow.

*2)* Generate $AS^3$ calculus terms describing the monitoring activities of the controller.

*3)* Generate $AS^3$ calculus terms describing the computation and optimization procedure of the total PI of all tasks monitored by the controller, and the system adaptation activities based on the total PI.

*4)* Compile the $AS^3$ calculus description of the controller into an executable agent using our $AS^3$ calculus to Java compiler [20].

In Step *1)*, the preconditions of each task are extracted to determine the *Readiness* function of the task. Similarly, other information, such as required types and minimum amounts of resources of a task *i*, and the latest completion time of a task *i*, is also extracted for functions $avail_i$ and $g_i$.

In Step *2),* our agent synthesis technique for situation awareness [20] is used to generate the $AS^3$ calculus terms describing the monitoring activities of the controller. These monitoring activities include determining the current situations, collecting available network bandwidth for calculating migration costs, and collecting the information for calculating resource availability.

For Step *3)*, the generation of $AS^3$ calculus terms describing the computation of the total PI and the system adaptation activities based on the total PI is easy. The computation of the total PI can be easily described using the internal computations provided in $AS^3$ calculus (see Section 3). The system adaptation activities can be described as invocations of system control services for resource management and agent migration, with parameters to be determined through the optimization of the total PI. The generation of $AS^3$ calculus terms describing the optimization procedure of the total PI is more difficult. Currently, a recursive process encoding the local search phase of GRASP (Greedy Randomized Adaptive Search Procedure) [30] described in $AS^3$ calculus will be generated to find a local maximum of the total PI. The construction phase of GRASP is omitted in our approach since we use the initial operations plan as a feasible solution to be constructed in the construction phase of GRASP.

## 8. Conclusions and Future Work

In this paper, we have presented a software cybernetics approach to scheduling and deploying workflows with timing and resource constraints in SBS. Our approach combines a logic-based technique for offline resource planning, and the synthesis of controllers for online monitoring and adaptation. The prototype of the constraint solver presented in Section 6 has been developed, and we are currently developing the controller generator based on the synthesis process presented in Section 7. Further research needs to be done, including the evaluation of our approach, and investigations on various optimization techniques in control theory to improve our approach. In addition, more in-depth investigations are needed to develop a unified framework for logic-based techniques and control theories in order to make this approach more useful.

## Acknowledgement

The work reported here was partially supported by the DoD/ONR under the Multidisciplinary Research Program of the University Research Initiative, Contract No. N00014-04-1-0723.

## References

[1] The Object Management Group (OMG), "Deployment and Configuration of Component-based Distributed Applications Specification", available at: http://www.omg.org/docs/ptc/03-07-02.pdf

[2] I. Ben-Shaul, O. Holder, and B. Lavva, "Dynamic Adaptation and Deployment of Distributed Components In Hadas," *IEEE Trans. on Software Engineering*, vol. 27(9), September 2001, pp. 769-787.

[3] M. W. Jang, and G. Agha. "Dynamic Agent Allocation for Large-Scale Multi-Agent Applications", *Proc. Int'l Workshop on Massively Multi-Agent Systems*, December 10-11, 2004, pp. 19-33.
[4] A. Liotta, G. Pavlou, and G. Knight, "Exploiting Agent Mobility for Large Scale Network Monitoring," *IEEE Network*, vol. 16(3), May 2002, pp. 7-15.
[5] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment," *J. ACM*, vol. 20(1), 1973, pp. 46-61.
[6] J. A. Stankovic and K. Ramamrithitham (Eds), *Hard Real-Time Systems*, IEEE Press, 1988.
[7] W. Zhao, K. Ramamritham and J. A. Stankovic, "Preemptive Scheduling Under Time and Resource Constraints," *IEEE Transactions on Computers,* vol. 36(8), 1987, pp 949-960.
[8] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS Negotiation in Real-Time Systems and its Application to Automatic Flight Control," *IEEE Real-Time Technology and Applications Symposium*, June 1997, pp 228-238.
[9] M. Caccamo, G. Buttazzo, and L. Sha, "Capacity Sharing for Overrun Control," *Proc. IEEE Real-Time Systems Symposium,* December 2000, pp 295-304.
[10] C. Lu, "Feedback Control Real-time Scheduling", Ph.D. thesis, University of Virginia, May 2001.
[11] C. Lu, J. A. Stankovic, G. Tao and S. H. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," *Real-Time Systems Journal*, vol. 23(1/2), 2002, pp. 85-126.
[12] P. Senkul, M. Kifer, and I. H. Toroslu, "A Logical Framework for Scheduling Workflows under Resource Allocation Constraints," *Proc. 28th Int'l Conf. on Very Large Data Bases (VLDB'02)*, 2002, pp. 694-705.
[13] H. Davulcu, M. Kifer, C.R. Ramakrishnan, and I.V. Ramakrishnan. "Logic based modeling and analysis of work flows". *Proc. ACM Symp. on Principles of Database Systems*, June 1998, pp. 25-33.
[14] L. Yang, J. M. Schopf and I. Foster, "Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments", *J. Parallel and Distributed Computing,* to appear, http://www-unix.mcs.anl.gov/~schopf/Pubs/Lingyun.Journal.july05.pdf.
[15] J. Yu and R. Buyya, "A Novel Architecture for Realizing GridWorkflow using Tuple Spaces", *Proc. 5th Int'l Workshop on Grid Computing,* November 2004, pp. 119-128.
[16] K. Keahey, K. Doering, and I. Foster, "From Sandbox to Playground: Dynamic Virtual Environments in the Grid", *Proc. 5th Int'l Workshop on Grid Computing,* November 2004, pp. 34-42.
[17] R. Hernandez, D. Vanderster and N. Dimopoulos, "Resource Management and Knapsack Formulations on the Grid", *Proc. 5th Int'l Workshop on Grid Computing,* November 2004, pp. 95-101.
[18] S. S. Yau, D. Huang, H. Gong, and H. Davulcu, "Situation-awareness for adaptable service coordination in service-based systems", *Proc. 29th Annual Int'l Computer Software and Application Conf. (COMPSAC' 05),* July 2005, pp.107-112.
[19] S. S. Yau, S. Mukhopadhyay, D. Huang, H. Gong, H. Davulcu, and L. Zhu, "Automated Agent Synthesis for Situation-Aware Service Coordination in Service-based Systems", technical report, Arizona State University, 2005, available at: http://dpse.eas.asu.edu/as3/papers/ASU-CSE-TR-05-009.pdf.
[20] S. S. Yau, H. Gong, D. Huang, W. Gao, and L. Zhu, "Automated Agent Synthesis for Situation Awareness in Service-based Systems," *Proc. 30th IEEE Int'l Computer Software and Applications Conf. (COMPSAC'06)*, September 2006, pp. 503-510.
[21] L. Cardelli and A. D. Gordon, "Mobile Ambients," *Theoretical Computer Science*, vol. 240(1), 2000, pp. 177-213.
[22] S. Ponnekanti and A. Fox, "Sword: A developer toolkit for web service composition", *11th Int'l World Wide Web Conf. (WWW 2002) Web Engineering Track*, 2002. Available at: http://www2002.org/CDROM/alternate/786/
[23] J. Rao, P. Kungas and M. Matskin, "Application of linear logic to web service composition", *Proc. 1st Int'l Conf. on Web Services (ICWS'03)*, 2003, pp. 3-9.
[24] E. Sirin, J. A. Hendler and B. Parsia, "Semi-automatic composition of web services using semantic descriptions", *Proc. Web Services: Modeling, Architecture and Infrastructure (WSMAI) Workshop in conjunction with the 5th Int'l Conf. on Enterprise Information Systems (ICEIS 2003),* 2003, pp. 17-24.
[25] K. Y. Cai, T.Y. Chen, and T.H. Tse, "Towards Research on Software Cybernetics", *Proc. 7th IEEE Int'l Symp. on High Assurance Systems Engineering (HASE'02)*, 2002, pp 240-241.
[26] K. Y. Cai, J. W. Cangussu, R. A. Decarlo, and A. P. Mathur, "An Overview of Software Cybernetics", *Proc. 11th Annual Int'l Workshop on Software Technology and Engineering Practice*, 2004, pp 77-86.
[27] K. Marriott and P. J. Stuckey, *Programming with Constraints: An Introduction*, MIT Press, 1998.
[28] T. Schrijvers, and D. S. Warren, "Constraint Handling Rules and Tabled Execution", *Proc. 20th Int'l Conf. on Logic Programming*, 2004, pp. 120-136.
[29] L. Ljung, *System Identification: Theory for the User (2nd Edition)*, Prentice Hall, 1998.
[30] M. Resende and C. Ribeiro, "Greedy Randomized Adaptive Search Procedures", *State of the Art Handbook in Metaheuristics*, F. Glover and G. Kochenberger (eds.), Kluwer publisher, 2002.