Modeling and Simulating Reconfigurable Networked Service Composites

Onyeka Ezenwoye
Computer and Information Sciences
Georgia Regents University
Augusta, Georgia 30904
Email: oezenwoye@gru.edu

S. Masoud Sadjadi
School of Computing and Information Sciences
Florida International University
Miami, Florida 33199
Email: sadjadi@cs.fiu.edu

Wei Wang
Department of Computer Science
San Diego State University
San Diego, California 92115
Email: wwang@mail.sdsu.edu

***Abstract*—Composite services are notoriously prone to failure, this is particularly true for long-running, and data-intensive services. Different composition strategies can be employed to make compositions robust. Any service composition strategy does impact performance at the lower network layer and needs to be assessed. Novel approaches are needed to model and evaluate dynamically reconfigurable service composition strategies. We propose an extensible framework to modeling and simulating the behavior of service compositions.**

Keywords: software framework, adaptive service composition, network simulation.

I. Introduction

We define a service as an application component that is remotely accessible to other applications and performs some useful activity (service) on their behalf. The service then is the behavior of the system, and is characterized by a set of states which include computation, communication, stored data, and physical condition. Service composition is achieved by defining executable abstractions that model the interaction between services. The complex nature of service compositions and the distributed and autonomous nature of their execution environments poses significant problems to building robust service-based applications. Services may fail due to problems in their environment or problems concerning the unmanaged nature of the communication channel. Accounting for all the dynamics of interactions of composite services is difficult. This is especially true for data-intensive service compositions where data movement and management is particularly problematic [1], [2,].

Compositions strategies need to meet various application requirements such as performance, and reliability. Composition decisions do impact multiple layers of the application stack, all the way to the lower network layer. The cost of composition strategies to overall performance of the system needs to be assessed. Due to the dynamic and autonomous nature of the services in a composition, the evaluation of composition strategies needs to be done in a controlled and repeatable manner by using simulation software. There are currently no tools for simulating flexible service compositions strategies in a way that allows for the nework performance impact of composition to be assessed. There is a need for the development of extensible frameworks for reconfigurable service composition and simulation. The framework should allow for the investigation of the relationship between composition strategy and overall performance of the system. A software framework provides libraries and software tools to organize and build applications in a specific application domain. The application domain in this case is networked service-based composites. The design of the framework should address problems at different layers of the application stack. The composition strategy of the service does have performance implications at the network layer because composition strategy determines how data is moved around the network. The performance impact on the network can be used to assess the suitability of the composition strategy.

In this paper, we present the ideas behind a software framework for modeling and evaluating dynamic service compositions. The framework should provide: (1) interface and coordination protocols for hybrid composition strategies, (2) extensible policy specification and algorithms for automatic adaptation at both pre-deployment and run time of the composites, (3) software simulator for modeling and monitoring service compositions, and execution with varying properties. A software simulator could build on existing open source network simulators such as NS-2 [3]. This tool should include components for modeling the behavior of atomic composable heterogeneous services whose behavior cover computation, communication and faults. It should permit modeling and monitoring of properties that include the type, speed and reliability of the network links. While allowing for the service compositions to be created graphically and then executed for evaluation. These tools will help in the exploration of the interplay between service composition strategies, and network properties. An extensible framework can easily evolve to simulate various application scenarios.

The rest of this paper is structures as follows; we articulate in Section II a service representation model. Section III we present a reconfigurable composition model. Section IV discusses the network layer. Related work and conclusion are in Sections V and VI respectively. We hope the ideas presented here will help lay the foundation for this service composition and simulation framework to support both wired and wireless networks, across multiple layers, in a data-centric fashion.

II. Service Representation

The structure of a composite facilitates its behavior and structurally the system can be seen as being composed of a set of interacting services. This definition is recursive since indeed each service can be another composite. However, the recursion stops when a service is considered to be *atomic* and its internal structure is not composite. A framework for modeling and simulation of service composition should permit the abstraction of atomic services. It should be possible to model attributes concerning:

1) Interprocess communication: Here, it should allow for algorithms to be defined to model communication methods from *point-to-point* to *broadcast* messaging over a network. It is important to model this behavior since it is needed to enable message exchange between reconfigurable services. Adaptable composition requires support for modifiable interprocess communication [4], [5].
2) Timing model: Here, it should allow for algorithms to be defined to model the timing of events in the service, related to computation and communication. In most distributed systems, execution proceeds asynchronously [6] and at the atomic level a service is inherently asynchronous. Thus the focus should be on an *asynchronous* timing model. However, abstractions could be defined in a way that allows for synchronous and partially synchronous timing behavior to be added.
3) Failure model: Here, it should be possible for algorithms to be defined to model faulty behavior which will include *transient* and *non-transient failures* in computation and communication mechanisms. This will also be useful since service instances and network partitions can be transient [4], [7].

Since the focus is on asynchronous timing, we model an atomic service as a combination of input-enabled process automata and channel automata [5], [6] that interact with each other and operate at varying speeds. The actions of the automata are classified as input, output or internal, where input and output are used for communication. Figure 1 illustrates the process and communication automaton with input and output actions.

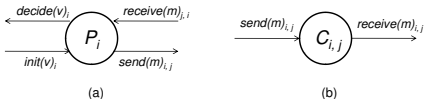

Fig. 1. I/O Automata, (a) Process automaton, (b) Channel automaton

For the process P_i, $init(v)_i$ represents the receipt of input value v, while $decide(v)_i$ represents a decision on v. The output action $send(m)_{i,j}$ represents process P_i sending a message m to P_j, while input action $receive(m)_{j,i}$ represents process P_i receiving a message from P_j. The channel automaton $C_{i,j}$ represents a FIFO message channel with input and output actions (*send* and *receive* respectively). When process and channel automata are combined, the output of one automaton is matched to the same-named input of the other automaton. The set of input and output actions represents the external *interface*. For each automaton, the *task* can be described as the set of actions performed, so $send(m)_{i,j}$ and $decide(v)_i$ are both regarded as single tasks for process P_i, while $receive(m)_{i,j}$ is thought of as a single task for channel $C_{i,j}$ [6]. For performance measurement at this level, the metric that could be used is task execution time. That is, the time it takes for a task to complete, in the absence of failure.

There is a need to not only define framework components that accurately combine process and channel behavior in an extensible manner, but also encapsulates distinctively the processes and message queue connector communication mechanisms for loosely coupled message communication.

III. Reconfigurable Composite Model

A composite service is an aggregation of tasks and the tasks are implemented by the integrated services. Each service is associated with one or more channels, as well as interfaces. The composite service itself is a service.

Composite services are described and published as service aggregators which are service providers themselves. A composite service models and coordinates the interactions between its constituent services. The two main service composition strategies are *orchestration* and *choreography* [9]. In orchestration, the interaction between services is coordinated in a *centralized* manner by a process which captures the logic of the interaction between the services. The logic of the interaction (control and data dependency) is encapsulated within the composite service.

Some service orchestration platforms do employ decentralized data flow techniques. This approach is typically utilized for data-intensive composition which is often seen in scientific workflow applications. In this model, integrated services also encapsulate data movement and management logic to facilitate the movement of data in a peer-to-peer manner. Here data staging tasks can be incorporated as part of the composition. Data staging allows for the data needed for computation tasks to be moved to different locations before and after execution. Control flow logic is retained by the composite thus issues with scalability persist. The composite service in the orchestration approach does constitute bottle-neck, and central point of failure, especially for data-intensive compositions [10] In contrast to the orchestration model, service choreography involves two or more services participating to provide service, in peer-to-peer manner. This model is characterized by decentralized control and data flow. Here, each service is aware of its role in the interaction and its immediate partner. This presents significant advantages to scalability and more efficient data flow. However, this model is difficult to manage especially in exceptional conditions. Dynamic reconfiguration is difficult since no service has a global view of the overall integration [11].

In order to support dynamic reconfiguration of service composites, a hybrid compostion model is needed. The hybrid model is a combination of orchestration and choreography, and overcomes some of the limitations of orchestration and

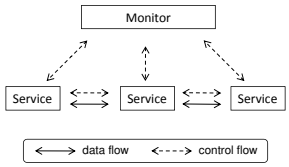

Fig. 2. A hybrid service composition model

choreography. Figure 2 is an illustration of the hybrid model. This model has a composite service, which we will refer to as the monitor. The monitor captures the logic of the interaction between the services participating in the choreography. Thus the monitor has the global view of the interaction between the integrated services [11], [12] This component intervenes only if necessary to provide required adaptability to the system. This monitor node observes the data and control-flow of the choreography. In case of failure, the monitor intervenes and for instance, assigns the load to another service and provides it with the information needed to resume the task. This monitor would encapsulate the behavioral and recovery policies that allow for the composition to be reconfigured. This hybrid model is scalable since multiple monitors can observe the same composition. When multiple monitors are used, the monitors would need to coordinate between themselves.

This hybrid sort of composition would support a system for the development and execution of dynamically reconfigurable service composites. Figure 3 shows an overview of the architecture of such an adaptive composition system. On the left side of Figure 3, the logic of the composite is specified. This specification should only be concerned with the data and control flow dependencies of the composition, and not crosscutting exceptional behavior.

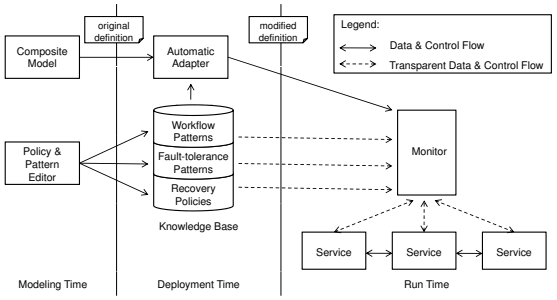

Fig. 3. Conceptual architecture of a reconfigurable composition system

During deployment time, the resulting composition is passed to an adapter, which in its turn automatically generates a functionally equivalent composition but with introduced redundancy for failure prone tasks. The automatic adapter will have an algorithm that identifies known composition patterns. Updated patterns will be stored in the Knowledge Base. Composition and Fault-tolerance software patterns, and recovery policy will be specified at a high-level. The generated composition (modified definition) is then deployed on the monitor. The monitor itself is a service that encapsulates the logic of the composition.

At run time, the monitor uses the composite definition as a global view of the interaction in order to observe the servicse participating in the integration. The monitor does not expose the functionality of the composition. The monitor will interact with the participant services via clearly defined interfaces. The monitor will implement pattern-matching algorithms that monitor the behavior of the composition and provides adaptive behavior when required. The algorithms should be based on the Recovery Policies, the Composition Patterns, and their corresponding Fault-Tolerant Patterns. For instance, dead-path elimination techniques [13] could be applied here in novel ways to achieve reconfiguration. The policy will be used to specify parameters for dead-path elimination. Since the patterns and policy are extensible, other recovery techniques such as retry and migration can be applied. The policy can also be used to specify parameters for those techniques.

This approach is modular and achieves separation of concerns since adaptive behavior is driven by externalized policies, patterns and monitors. A framework for modeling and simulating of compositions would allow for various composition stratgies and scenarios to be evaluated in a repeatable manner. The framwork would support the modeling of the service interactions by permitting the definition of the sequence of execution of the interaction services, as well as the definition of the channels through which communication between services is achieved. These definitions can be created as separate but related direct graph models. Some research has shown that the specification of a composite service can be declaratively modeled using finite-state automaton [14], [15].

The separation of interaction sequence and communication channel representation is useful so that the models can be simplified while allowing for flexible combinations of choreography and orchestration to be achieved with composability of communication mechanisms. Knowledge from existing standards for specification of Web service composition can be leveraged. For performance monitoring of composite services, the overall task completion time (*makespan*) of the composite service can be used as a metric. It should be possible to calculate the makespan since task completion time is also measured at the atomic service level. The makespan is directly related to the performance of the network who's peformance metrics can be observed as discussed in the next section IV.

IV. Network Simulation

To simulate the composition and its effects on the network, the framework can leverage existing, and widely used network simulators such as, NS-2 [17]. NS-2 is a discrete event network simulator that focuses on the simulation of IP networks at the packet level. Simulation occurs by translating physical activities to events. These events are queued and processed in the order of their scheduled occurrences, and time progresses as the events are processed. NS-2 is widely accepted by the research community, as it supports many protocols such as TCP and UDP, router queuing policies, Multicast and Unicast transport, Multimedia, wired and wireless protocols.

The simulator is a collection of open source object-oriented network simulation libraries. The implementation language for NS-2 is C++.

NS-2 is sophisticated enough to model the network and capture the desired service attributes of the communication, timing and failure (described in Section II). It should be possible for the framework to map the model of the composite service directly onto the NS-2 class library. For performance monitoring, the metrics available from NS-2 include:

- Number of data packets sent.
- Number of data packets received by the destination host.
- Total number of routing packets.
- Packet delivery fraction - ratio of received packets over sent packets in percentage.
- End to end delay - average time for a data packet delivered from host to destination.

These metrics can be used to assess the overall performance of the system. The performance of the compostion can be measured as its task completion times (discussed in section III)

V. Related Work

Current tools for modeling and simulation such as Activiti [18] have largely ignored the relationship between service composition strategy and the underlying network. Traditionally the research on service composition is focused on application layer. Simulators such as GridSim [19], SimGrid [20], Bricks [21], and MicroGrid [22] focus mainly on the resource allocation problem in the management domain of the interacting services. A lot of emphasis is put on schedulers for data-intensive applications. These simulators are not general enough to support different application areas and composition strategies. They don't permit interactions to be modeled graphically, nor the separation of interaction logic from communication channels. Also, no considerations are made for the network layer, and failure models are not captured. These simulators cannot be extended to address these issues.

Accurate simulation of network performance has been studied in the past decades with several existing simulation tools such as Qualnet [23], and NS-2 [17]. None of these simulation tools allow the simulation of service composition. None of the above simulation platforms consider service composition at application layer and the complex network interaction at lower layers. In a dynamic networked service-based system, high level composition decision are closely associated to low-level network properties. A desirable framework should consider both service composition and network dynamics jointly, establishing a novel approach to simulating composite service-oriented systems.

VI. Conclusion

Service composition has inate challenges due to its interaction model, networked communication, and heterogeneous execution environments. Composition decisions do impact multiple layers of the application stack, all the way to the lower network layer and thus should not be treated in isolation. We propose a novel simulation framework for evaluating the behavior of service compositions. This framework will allow the investigation of strategies to achieve scalable and reconfigurable composition, while exploring the relationship between data movement and service composition schemes.

Acknowledgment

This work is supported in part by the NSF under Grant Numbers of I/UCRC IIP-1338922, and AIR IIP-1237818.

References

[1] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," in *IEEE International Conference on Data Engineering*, 2009.

[2] G. Kandaswamy, A. Mandal, and D. Reed, "Fault tolerance and recovery of scientific workflows on computational grids," in *8th IEEE International Symposium on Cluster Computing and the Grid*, 2008.

[3] "Ns-2," http://www.isi.edu/nsnam/ns/.

[4] A. Nguyen-Tuong, "Integrating fault-tolerance techniques in grid applications," Ph.D. dissertation, School of Engineering and Applied Science, University of Virginia, United States, August 2000.

[5] R. Milner, *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, June 1999.

[6] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[7] V. P. Nelson, "Fault-tolerant computing: Fundamental concepts." *IEEE Computer*, vol. 23, no. 7, pp. 19–25, 1990.

[8] D. Riehle, "Composite design patterns," in *Proceedings of International Conference on Object-Oriented Programming Systems, Languages and Applications*. ACM Press, 1997.

[9] C. Peltz, "Web services orchestration and choreography," *IEEE Computer*, vol. 36, no. 10, pp. 44–52, 2003.

[10] E. Deelman and A. Chervenak, "Data management challenges of data-intensive scientific workflows," in *8th IEEE International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2008.

[11] O. Ezenwoye and B. Tang, "Monitoring decentralized interacting web services with a global state choreography model," in *IEEE International Conference on Web Services*. IEEE Computer Society, 2010.

[12] A. Cass, B. Lerner, E. McCall, L. Osterweil, and A. Wise, "Logically central, physically distributed control in a process runtime environment," Department of Computer Science, University of Massachusetts, Tech. Rep. Technical Report UM-CS-1999-065, 1999.

[13] M. Weidlich, A. Grokopf, and A. P. Barros, "Realising dead path elimination in BPMN," in *IEEE Conference on Commerce and Enterprise Computing*. IEEE Computer Society, 2009, pp. 345–352.

[14] M. H. T. Beek, A. Bucchiarone, and S. Gnesi, "Formal methods for service composition," *Annals of Mathematics, Computing and Teleinformatics*, vol. 1, no. 5, 2007.

[15] S. Nanz and T. Tolstrup, "Goal-oriented composition of services," in *7th international conference on Software composition*. Springer, 2008.

[16] Z. Qiu, X. Zhao, C. Cai, and H. Yang, "Towards the theoretical foundation of choreography," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007.

[17] T. Issariyakul, E. Hossain, T. Issariyakul, and E. Hossain, *Introduction to Network Simulator 2 (NS2)*. Springer US, 2009.

[18] T. Rademakers, *Activiti in Action : Executable business processes in BPMN 2.0*, 1st ed. Manning Publications, 2012.

[19] R. Buyya and M. Murshed, "Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1175–1220, 2002.

[20] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *10th International Conference on Computer Modeling and Simulation*. IEEE Computer Society, 2008.

[21] K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, and U. Nagashima, "Performance evaluation model for scheduling in global computing systems," *Int. J. High Perform. Comput. Appl.*, vol. 14, August 2000.

[22] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien, "The microgrid: a scientific tool for modeling computational gridsr," in *ACM/IEEE Conference on Supercomputing*, 2000.

[23] "Qualnet2," http://www.scalable-networks.com/.