Analyzing Interactions of Asynchronously Communicating Software Components
(Invited Paper)

Tevfik Bultan

Department of Computer Science
University of California, Santa Barbara
bultan@cs.ucsb.edu

Abstract. Since software systems are becoming increasingly more concurrent and distributed, modeling and analysis of interactions among their components is a crucial problem. In several application domains, message-based communication is used as the interaction mechanism, and the communication contract among the components of the system is specified semantically as a state machine. In the service-oriented computing domain this type of message-based communication contracts are called "choreography" specifications. A choreography specification identifies allowable ordering of message exchanges in a distributed system. A fundamental question about a choreography specification is determining its realizability, i.e., given a choreography specification, is it possible to build a distributed system that communicates exactly as the choreography specifies? In this short paper we give an overview of this problem, summarize some of the recent results and discuss its application to web service choreographies, Singularity OS channel contracts, and UML collaboration (communication) diagrams.

1 Introduction

Nowadays, many software systems consist of multiple components that execute concurrently on different machines that are physically distributed, and interact with each other through computer networks. Moreover, new trends in computing, such as service-oriented architecture, cloud computing, multi-core hardware, wearable computing, all point to even more concurrency and distribution among the components of software systems in the future. Concurrent and distributed software systems are increasingly used in every aspect of society and in some cases provide safety critical services. Hence, it is very important to develop techniques that guarantee that these software systems behave according to their specifications.

A crucial problem in dependability of concurrent and distributed software systems is the coordination of different components that form the whole system. In order to complete a task, components of a software system have to coordinate their executions by interacting with each other. One fundamental question is,

what should be the interaction mechanism given the trend for increased level of concurrency and distribution in computing? One emerging paradigm is message-based communication [2, 7, 10, 8], where components interact with each other by sending and receiving messages. Given these trends, we conclude that analyzing message-based interactions among software components is a timely and significant research problem.

2 Specification of Message-Based Interactions

Specification and analysis of message-based interactions has been an active research area studied in several application domains including coordination in service-oriented computing [7, 12], interactions in distributed programs [1] and process isolation at the OS level [8].

Service oriented computing provides technologies that enable multiple organizations to integrate their businesses over the Internet. Typical execution behavior in this type of distributed systems involves a set of autonomous peers interacting with each other through messages. Modeling and analysis of interactions among the peers is a crucial problem in this domain due to following reasons: 1) Organizations may not want to share the internal details of the services they provide to other organizations. In order to achieve decoupling among different peers, it is necessary to specify the interactions among different services without referring to the details of their local implementations. 2) Modeling and analyzing the global behavior of this type of distributed systems is particularly challenging since no single party has access to the internal states of all the participating peers. Desired behaviors have to be specified as constraints on the interactions among different peers since the interactions are the only observable global behavior. Moreover, for this type of distributed systems, it might be worthwhile to specify the interactions among different peers before the services are implemented. Such a top-down design strategy may help different organizations to better coordinate their development efforts.

Choreography languages enable specification of such interactions. A choreography specification corresponds to a global ordering of the message exchange events among the peers participating to a composite service, i.e., a choreography specification identifies the set of allowable message sequences for a composite web service.

3 Choreography Analysis

Specification of interactions in a software system as a choreography leads to several interesting research problems [6]:

- *Realizability:* Given a choreography specification, determining if there exists a set of components that generate precisely the set of message sequences specified by the choreography specification.

- *Synthesis:* Given a choreography specification, synthesizing a set of components that generate precisely the set of message sequences specified by the choreography specification.
- *Conformance:* Determining if a set of given components adhere to a given choreography specification.
- *Synchronizability:* Determining if the set of interactions generated by a given set of components remain the same under asynchronous and synchronous communication.

Some formalizations of these questions lead to unsolvable problems. For example, choreography conformance problem is undecidable when asynchronous communication is used. This is because, systems where peers communicate asynchronously with unbounded FIFO message queues can simulate Turing Machines [5].

It is important to note that the choreography analysis problem is not isolated to the area of service-oriented computing. It is a fundamental problem that appears in any area where message-based communication is used to coordinate interactions of multiple concurrent or distributed components. For example, recently, earlier results on choreography analysis have been applied to analysis of Singularity channel contracts [11]. Singularity is an experimental operating system developed by Microsoft Research in order to improve the dependability of software systems [9]. In the Singularity operating system all inter-process communication is done via messages sent through asynchronous communication channels. Each channel is governed by a channel contract [8]. A channel contract is basically a state machine that specifies the allowable ordering of messages between the client and the server. Hence, channel contracts serve the same purpose that choreography specifications serve in service oriented computing.

4 Recent Results

There has been some recent progress in addressing these research problems. It has been shown that synchronizability checking is decidable [3]. It can be solved by comparing the behavior of a system with synchronous communication to the behavior of the same system with bounded asynchronous communication where the queue sizes are limited to one. This result also leads to effective approaches to choreography conformance checking. Although choreography conformance problem is undecidable in general, synchronizability analysis identifies a class of systems for which choreography conformance can be checked using synchronous communication instead of asynchronous communication. This means that message queues can be removed during the conformance analysis, significantly reducing the state space of the analyzed system.

More recently, it has been shown that choreography realizability problem is decidable for systems communicating with asynchronous messages using unbounded FIFO message queues [4]. This also means that for realizable choreography specifications, synthesis problem can be solved by projecting the given choreography specification to each component that participates to the choreography.

References

1. J. Armstrong. Getting Erlang to Talk to the Outside World. In *Proc. ACM SIGPLAN Work. on Erlang*, pages 64–72, 2002.
2. G. Banavar, T. Deepak Chandra, R. E. Strom, and D. C. Sturman. A Case for Message Oriented Middleware. In *Proceedings of the 13th Int. Symp. Distributed Computing*, pages 1–18, 1999.
3. S. Basu and T. Bultan. Choreography Conformance via Synchronizability. In *Proc. 20th Int. World Wide Web Conf.*, 2011.
4. S. Basu, T. Bultan, and Meriem Ouederni. Deciding Choreography Realizability. In *Proc. 39th Symp. Principles of Programming Languages*, 2012.
5. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.
6. T. Bultan, X. Fu, and J. Su. Analyzing conversations: Realizability, synchronizability, and verification. In Luciano Baresi and Elisabetta Di Nitto, editors, *Test and Analysis of Web Services*, pages 57–85. Springer, 2007.
7. M. Carbone, K. Honda, N. Yoshida, R. Milner, G. Brown, and S. Ross-Talbot. A Theoretical Basis of Communication-Centred Concurrent Programming, W3C Note, October 2006. http://www.w3.org/2002/ws/chor/edcopies/theory/note.pdf.
8. M. Fähndrich, M. Aiken, C. Hawblitzel, O. Hodson, G. C. Hunt, J. R. Larus, and S. Levi. Language support for fast and reliable message-based communication in singularity os. In *Proc. 2006 EuroSys Conf.*, pages 177–190, 2006.
9. G. C. Hunt and J. R. Larus. Singularity: rethinking the software stack. *Operating Systems Review*, 41(2):37–49, 2007.
10. D. A. Menascé. Mom vs. rpc: Communication models for distributed applications. *IEEE Internet Computing*, 9(2):90–93, 2005.
11. Z. Stengel and T. Bultan. Analyzing Singularity Channel Contracts. In *Proceedings of the 18th International Symposium on Software Testing and Analysis*, pages 13–24, 2009.
12. Web Service Choreography Description Language (WS-CDL). http://www.w3.org/TR/ws-cdl-10/, 2005.