ERCIM “Alain Bensoussan” Fellowship Scientific Report

Fellow: Aliaksandr Lazovik
Visited Location : CWI (NL), INRIA (FR)
Duration of Visit: 01/04/2007 – 30/09/2008

I - Scientific activity

(1 page at maximum)

My scientific activities mainly consist of two directions: (i) service composition and related issues, especially in the context of coordination Reo language (CWI part), and (ii) service monitoring and diagnosability, optimal observability w.r.t. diagnosability and system repair (INRIA part).

In CWI I was working with coordination language Reo, and its application to service composition. My primary task here was to consider the possibility of applying Reo language to coordination of services, and to guide the implementation of supporting tools and framework (together with a couple of recently applied PhD students, partially under my guidance). The tools have been implemented as a set of Eclipse plug-ins [8]. Personally, I consider Reo as a language that has a number of very interesting and promising ideas, that other process languages lack. For example, Reo is a declarative language, that is, each channel may represent not only actual control or data flow, but also an arbitrary constraint between two execution nodes. Thus, we encoded the Reo channels as pure constraint satisfaction problem and used a java-based constraint solver to implement the Reo execution engine [3]. Another nice feature is compositionality of the language, that allows easily introduce new domain-specific channels and connectors or re-use previously defined process definitions as a part of a new process [7]. The need for dynamic reconfiguration results in a set of algorithms for customization of Reo model on the fly [9].

As for my second hosting institute (INRIA), I was working at the diagnosability and issues of services and their corresponding theoretical models. The diagnosability property recognizes if a system model can be unambiguously diagnosable; that is, if all faults can be detected using only the information given by the observable events. Usually, large and complex systems require an automatic fault detection and isolation, but to specify the minimal observability degree of a system to be diagnosable is not a trivial task. During my stay at INRIA we found the necessary and sufficient conditions for observability (the list of observable events) that a system has to maintain to be diagnosable. We concentrated on two problems: first, we transformed a diagnosable system into one with minimal degree of observability and still diagnosable. Second, we transformed a non-diagnosable system into diagnosable by increasing the degree of observability. We also expanded the developed algorithms with several extensions: for distinguishability, for predictability and for extended fault models [2, 4].

Additionally to the above mentioned research directions, I also studied the application of mash-up techniques to enterprise service computing [1, 5], as well as the issues of business process customization [6].

II- Publication(s) during your fellowship

Please insert the title(s), author(s) and abstract(s) of the published paper(s). You may also mention the paper(s) which were prepared during your fellowship period and are under reviewing.

2008

1. Z. Maraikar, A. Lazovik. Building Mashups for the Enterprise with SABRE. *Int. Conf on Service-Oriented Computing (ICSOC-08)*, 2008.

   Abstract: The explosive popularity of mashups has given rise to a plethora of web-based tools for rapidly building mashups with minimal programming effort. In turn, this has spurred interest in using these tools to empower end-users to build situational applications for business. Situational applications based on Reo (SABRE) is a service composition platform that addresses service heterogeneity as a first-class concern by adopting a mashup's data-centric approach. Built atop the Reo coordination language, SABRE provides tools to combine, filter and transform web services and data sources like RSS and ATOM feeds. Whereas other mashup platforms intermingle data transformation logic and I/O concerns, we aim to clearly separate them by formalising coordination logic within a mashup. Reo's well-defined compositional semantics opens up the possibility of constructing a mashup's core logic from a library of pre-built connectors. Input/output in SABRE is handled by service stubs generated by combining a syntactic service specification such as WSDL with a constraint automaton specifying service behavior. These stubs insulate services from misbehaving clients while protecting clients against services that do not conform to their contracts. We believe these are compelling features as mashups graduate from curiosities on the Web to situational applications for the enterprise.

2. L. Brandan-Briones, A. Lazovik, and P. Dague. Optimal observability for diagnosability. *International Workshop on Principles of Diagnosis (DX-08)*, 2008.

   Abstract: The diagnosability property recognizes if a system model can be unambiguously diagnosable; that is, if all faults can be detected using only the information given by the observable events. Usually, large and complex systems require an automatic fault detection and isolation, but to specify the minimal observability degree of a system to be diagnosable is not a trivial task.

   In this paper we give the necessary and sufficient conditions for observability (the list of observable events) that a system has to maintain to be diagnosable. We concentrate on two problems: first, we transform a diagnosable system into one with minimal degree of observability and still diagnosable. Second, we transform a non-diagnosable system into diagnosable by increasing the degree of observability. We also expand the developed algorithms with several extensions: for distinguishability, for predictability and for extended fault models.

3. D. Clarke, J. Proenca, A. Lazovik, and F. Arbab. Deconstructing Reo. *Int. Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA-08)*, 2008.

   Abstract: Coordination in Reo emerges from the composition of the behavioural constraints of the primitives, such as channels, in a component connector. Understanding and implementing Reo, however, has been challenging due to interaction of the channel metaphor, which is an inherently local notion, and the non-local nature of constraint propagation imposed by composition. In this paper, the channel metaphor takes a back seat, and we focus on the behavioural constraints imposed by the composition of primitives, and phrase the semantics of Reo as a constraint satisfaction problem. Not only does this provide a clear intensional description of the behaviour of Reo connectors in terms of synchronisation and data flow

constraints, it also paves the way for new implementation techniques based on constraint propagation and satisfaction. In fact, decomposing Reo into constraints provides a new computational model for connectors, which we extend to model interaction with an unknown external world beyond what is currently possible in Reo

4. L. Brandan-Briones, A. Lazovik, and P. Dague. Optimizing the system observability level for diagnosability. *Int. Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA-08)*, 2008.

   Abstract: The diagnosability property recognizes if a system model can be unambiguously diagnosable; that is, if all faults can be detected using only the information given by the observable events. Usually, large and complex systems require an automatic fault detection and isolation, but to specify the minimal observability degree of a system to be diagnosable is not a trivial task. In this paper we give the necessary and sufficient conditions for observability (the list of observable events) that a system has to maintain to be diagnosable. We concentrate on two problems: first, we transform a diagnosable system into one with minimal degree of observability and still diagnosable. Second, we transform a non-diagnosable system into diagnosable by increasing the degree of observability. We also expand the developed algorithms with several extensions: for distinguishability, for predictability and for extended fault models.

5. Z. Maraikar and A. Lazovik. ReforMing Mashups. *Young Researchers Workshop on Service-Oriented Computing (YRSOC-08)*, 2008.

   Abstract: The explosive popularity of mashups has given rise to a plethora of ``mashup platforms". Using these web-based tools, mashups can be rapidly constructed with minimal programming effort. Reo for Mashups (ReforM) is a service composition platform that addresses service heterogeneity as a first-class concern, by adopting a mashup's data-centric approach. Built atop the Reo coordination language, ReforM provides tools to combine, filter and transform web services and data sources like RSS and ATOM feeds. Whereas other mashup platforms intermingle data transformation logic and I/O concerns, we aim to clearly separate them by formalising the coordination within a mashup. Reo's well-defined compositional semantics opens up the possibility of constructing a mashup's core from a library of prebuilt connectors. We believe these are compelling features as mashups graduate from curiosities on the Web to situational applications for the enterprise.

2007

6. A. Lazovik and H. Ludwig. Managing Process Customizability and Customization: Model, Language and Process. In *Int. Conf. on Web Information Systems Engineering (WISE-07)*, 2007.
   Abstract: One of the fundamental ideas of services and service oriented architecture is the possibility to develop new applications by composing existing services into business processes. However, only little effort has been devoted so far to the problem of maintenance and customization of already composed processes. In the context of global service delivery, where process is delivered to clients, it is critical to have a possibility to customize the standardized reference process for each particular customer. Having a standardized delivery process yields many benefits: interchangeable delivery teams, enabling 24/7 operations, labor cost arbitrage, specialization of delivery teams, making knowledge shared between all customers, optimization of standardized processes by re-engineering and automation. In the paper we propose an approach, where reference process models are used explicitly in the process lifecycle, where customer-specific process instantiations are obtained by a series of customization steps over reference processes. To show the feasibility of the approach, we developed a process-independent language to express different customization options for the reference business

processes. We also provided an implementation that extends WebSphere BPEL4WS Editor to introduce process customizations to BPEL4WS processes.

7. A. Lazovik and F. Arbab. Using Reo for Service Coordination. In *Int. Conf. on Service-Oriented Computing (ICSOC-07)*, LNCS 4749, pages 392--397, Springer, 2007.
   Abstract: In this paper we address coordination of services in complex business processes. As the main coordination mechanism we rely on a channel-based exogenous coordination language, called Reo, and investigate its application to service-oriented architectures. Reo supports a specific notion of composition that enables coordinated composition of individual services, as well as complex composite business processes. Accordingly, a coordinated business process consists of a set of web services whose collective behavior is coordinated by a Reo expression.

   In this approach, it is easy to maintain a loosely coupled environment with services knowing nothing about each other. Although it is claimed that BPEL-like languages maintain service independence, in practice they hard-wire services through the connections that they specify in the process itself. In contrast, Reo allows us to concentrate only on important protocol decisions and define only those restrictions that actually form the domain knowledge, leaving more freedom for process specification and choice of individual services compared to traditional approaches.

8. C. Koehler, A. Lazovik, and F. Arbab. ReoService: Coordination Modeling Tool. In *Int. Conf. on Service-Oriented Computing (ICSOC-07)*, LNCS 4749, pages 625--626, Springer, 2007.
   Abstract: Coordination in SOA addresses dynamic topologies of interactions among services. Most efforts up to now have been focused on statically defined composition of services, e.g., using BPEL. To the best of our knowledge, there are no serious means to address the issues of dynamic coordination to accommodate continuously changing requirements. While BPEL is a powerful standard for service composition, it lacks support for typical coordination constraints, like synchronisation, mutual exclusion, and context-dependency.

   In this paper we present ReoService, which is a modeling tool for coordinating business processes. ReoService is based on Reo – a general framework for coordinating components in distributed systems. Reo is a channel-based exogenous coordination language wherein complex coordinators, called connectors, are compositionally built out of simpler ones. The simplest connectors are a set of user-defined communication channels with well-defined behavior. The emphasis in this model is on connectors, not on the services to connect. In this sense, ReoService acts as a “glue” language that interconnects and coordinates services in a distributed business process.

9. C. Koehler, A. Lazovik, and F. Arbab. Connector Rewriting with High-Level Replacement Systems. In FOCLASA-07, 2007.
   Abstract: Reo is a language for coordinating autonomous components in distributed environments. Coordination in Reo is performed by circuit-like connectors, which are constructed from primitive channels with well-defined behavior. These channels are mobile, i.e. can be dynamically created and reconfigured at run-time. Based on these language features, we introduce a high-level transformation system for Reo. We show how transformations of Reo connectors can be defined using the theory of high-level replacement (HLR) systems. This leads to a powerful notion of dynamic connector reconfiguration in Reo. Moreover, the rewrite rules are naturally expressed in Reo's visual syntax for connectors.

   Applications of this framework are manifold, due to the generality of the field of coordination. In this paper we provide an example from the area of Service-oriented Computing.

III -Attended Seminars, Workshops, and Conferences

Please identify the name(s), date(s) and place(s) of the events in which you participated during your fellowship period.

Lab visits:

November 27-30, 2007 Tutorial on Reo Coordination Tools. Technical University of Vienna, (invited by Prof. Scharam Dustdar)

November 20, 2007 Discussion on planning using constraint programming. Groningen, (invited by Prof. Marco Aiello)

Talk deliveries:

January 11, 2007 Using Reo for Service Coordination. INRIA Saclay, Parc Orsay Universite (FR)

December 21, 2007 Using Reo for Service Coordination. Rijksuniversiteit, Groningen (NL)

December 3-7, 2007 Managing Process Customizability and Customization: Model, Language and Process. In Int. Conf. on Web Information Systems Engineering (WISE-07), Nancy (FR)

September, 2007 Using Reo for Service Coordination. In Int. Conf. on Service-Oriented Computing (ICSOC-07), Vienna

April 3, 2007 Service request languages based on planning, IPA Lentedagen on Service-oriented Computing, Heeze (NL)

IV – Research Exchange Programme (12 month scheme)

Please identify the name(s), date(s) and place(s) of your Research Exchanges during your fellowship period and detail them .