Chapter 12
The Assurance Point Model for Consistency and Recovery in Service Composition

Susan D. Urban
Texas Tech University, USA

Le Gao
Texas Tech University, USA

Rajiv Shrestha
Texas Tech University, USA

Yang Xiao
Arizona State University, USA

Zev Friedman
Texas Tech University, USA

Jonathan Rodriguez
Texas Tech University, USA

ABSTRACT

This research has defined an abstract execution model for establishing user-defined correctness and recovery in a service composition environment. The service composition model defines a hierarchical service composition structure, where a service is composed of atomic and/or composite groups. The model provides multi-level protection against service execution failure by using compensation and contingency at different composition granularity levels. The model is enhanced with the concept of assurance points (APS) and integration rules, where APSs serve as logical and physical checkpoints for user-defined consistency checking, invoking integration rules that check pre and post conditions at different points in the execution process. The unique aspect of APSs is that they provide intermediate rollback points when failures occur, thus allowing a process to be compensated to a specific AP for the purpose of rechecking pre-conditions before retry attempts. APSs also support a dynamic backward recovery process, known as cascaded contingency, for hierarchically nested processes in an attempt to recover to a previous AP that can be used to invoke contingent procedures or alternate execution paths for failure of a nested process. As a result, the assurance point approach provides flexibility with respect to the combined use of backward and forward recovery options. Petri Nets have been used to define the semantics of the assurance point approach to service composition and recovery. A comparison to the BPEL fault handler is also provided.

DOI: 10.4018/978-1-61350-104-7.ch012
INTRODUCTION

In a service-based architecture, a process is composed of a series of calls to distributed Web services and Grid services that collectively provide some specific functionality of interest to an application (Singh & Huhns, 2005). In a traditional, data-oriented, distributed computing environment, a distributed transaction is used to provide certain correctness guarantees about the execution of a transaction over distributed data sources. In particular, a traditional, distributed transaction provides all-or-nothing behavior by using the two-phase commit protocol to support atomicity, consistency, isolation, and durability (ACID) properties (Kifer, Bernstein, & Lewis, 2006). A process in a service-oriented architecture, however, is not a traditional ACID transaction due to the loosely-coupled, autonomous, and heterogeneous nature of the execution environment. When a process invokes a service, the service performs its function and then terminates, without regard for the successful termination of the global process that invoked the service. If the process fails, reliable techniques are needed to either 1) restore the process to a consistent state or 2) correct critical data values and continue running.

Techniques such as compensation and contingency have been used as a form of recovery in past work with transactional workflows (e.g., Worah & Sheth, 1997) and have also been introduced into recent languages for service composition (e.g., Lin & Chang, 2005). In the absence of a global log file, compensation provides a form of backward recovery, executing a procedure that will “logically undo” the effects of completed and/or partially executed operations. Contingency is a form of forward recovery, providing an alternate execution path that will allow a process to continue execution. Some form of compensation may be needed, however, before the execution of contingency plans. Furthermore, nested service composition specifications can complicate the use of compensating and contingent procedures. To provide a reliable service composition mechanism, it is important to fully understand the semantics and complementary usage of compensation and contingency, as well as how they can be used together with local and global database recovery techniques and nested service composition specifications. Service composition models also need to be enhanced with features that allow processes to assess their execution status to support more dynamic ways of responding to failures, while at the same time validating correctness conditions for process execution.

This research has defined an abstract execution model for establishing user-defined correctness and recovery in a service composition environment. The research was originally conducted in the context of the DeltaGrid project, which focused on building a semantically-robust execution environment for processes that execute over Grid Services (Xiao, 2006; Xiao, Urban, & Dietrich, 2006; Xiao, Urban, & Liao, 2006; Xiao & Urban, 2008). The service composition model defines a hierarchical service composition structure, where a service is composed of atomic and/or composite groups. An atomic group is a service execution with optional compensation and contingency procedures. A composite group is composed of two or more atomic and/or composite groups and can also have optional compensation and contingency procedures. A unique aspect of the model is the provision for multi-level protection against service execution failure by using compensation and contingency at different composition granularity levels, thus maximizing the potential for forward recovery of a process when failure occurs. The work in (Xiao and Urban, 2009) presents the full specification of the model using state diagrams and algorithms to define the semantics of compensation and contingency in the recovery process.

Our more recent work has extended the DeltaGrid service composition and recovery model in (Xiao et al., 2009) with the concept of Assurance Points (APs) and integration rules to provide a more flexible way of checking constraints and...