Ensuring composite services reliability is a challenging problem. Indeed, due to the inherent autonomy and heterogeneity of Web services, it is difficult to predict and reason about the behavior of the overall composite service (CS). Generally, previous approaches develop, using their modeling formalisms, a set of techniques to analyze the composition model and check “correctness” properties. Although powerful, these approaches may fail in some cases in order to ensure CS-reliable executions, even if they formally validate its composition model. This is because properties specified in the studied composition model remain assumptions that may not coincide with the reality (i.e., effective CS executions). Sharing the same issue, we present a reengineering approach that starts from CS executions log to improve its recovery mechanisms. Basically, we propose a set of mining techniques to discover CS transactional behavior from an event-based log. Then, based on this mining step, we use a set of rules in order to improve its reliability.

Keywords: mining; reliability; transactional Web service; Web services compositions

INTRODUCTION
Nowadays, enterprises are able to outsource their internal business processes as services and make them accessible via the Web. Then they can dynamically combine individual services to provide new value-added composite services (CS). Due to the inherent autonomy and heterogeneity of Web services, a fundamental problem concerns the guarantee of correct executions of a CS. An execution is correct if it reaches its objectives or fails (properly) according to the designer’s requirements.

• **Motivating example:** Let us consider an application for an online travel arrangement carried out by a CS, as illustrated in Figure 1. The customer specifies his or her requirements in terms of destination
and hotel through the CRS service. The application launches in parallel flight and hotel reservation (FR and HR, respectively) after a study of the local transport accommodations (LTA). The ADC service disposes administrative documents. Then the customer is requested to pay by credit card (PCC), pay by check (PCh), or pay by TIP (PTIP). The Send Documents (SD) service ensures the delivery of documents to the customer. To deal with exceptions, designers specify additional mechanisms for failures handling and recovery. First, they specify that the hotel reservation can be compensated (e.g., by cancellation) when the FR service fails to reserve a flight, and reciprocally. Second, to ensure the payment, they specify the PCh service as a payment alternative for the PCC service. Similarly, they specify the PTIP service as a payment alternative for the PCh service with the assumption that the PTIP service always succeeds. Finally, designers specify that CRS, LTA, ADC, and SD services are sure to complete. The main problem at this stage is how to ensure that the specified CS model guarantees reliable executions.

Generally, previous approaches develop, based on their modeling formalisms, a set of techniques to analyze the composition model and check “correctness” properties. Although powerful, these approaches may fail in some cases in order to ensure CS reliable executions, even if they formally validate the CS model. This is because properties specified in the studied composition models remain assumptions that may not coincide with reality.

Back to our example, let us suppose for instance, that in reality (by observation of sufficient execution cases), the FR and PCh services never fail and the PTIP service is not sure to complete. That means, among other things, that (i) there is no need for the HR service to support compensation policies, which can be costly; and (ii) the payment can fail while the hotel and flight reservations are maintained. Formal approaches cannot deal with such anomalies.

Mining the effective transactional behavior allows detection of gaps, mentioned previously, and improvement of the application reliability. For instance, in our example, mining the transactional behavior allows improvement of the CS model by specifying the PCh service as a payment alternative for the PTIP service (since we notice that PCh is sure to complete).

**Overview of our approach:** As explained in section 2, we distinguish between the control flow and the transactional flow of a composite service. The control flow specifies its execution logic (without undesired failures), while the transactional flow defines its recovery mechanisms.

In this article, we present an approach to improve CS recovery mechanisms based on the...
Security System for Distributed Business Applications
www.igi-global.com/article/security-system-distributed-business-applications/3056?camid=4v1a