Service Interaction Patterns:

Towards a Reference Framework for Service-based Collaborative Business Processes

Alistair Barros
SAP Research Centre Brisbane, Australia
alistair.barros@sap.com

Marlon Dumas and Arthur ter Hofstede
Queensland University of Technology, Australia
m.dumas@qut.edu.au

Service-oriented Computing (SOC) is emerging as a promising paradigm for integrating software applications within and across organizational boundaries. In this paradigm, independently developed and operated applications are exposed as (Web) services which are then interconnected using a collection of Web-based standards including SOAP, WS-Addressing, WSDL, etc. While the technology for developing basic services and interconnecting them on a point-to-point basis has attained a certain level of maturity, there are many open challenges when it comes to managing interactions involving large numbers of services and occurring in the setting of collaborative and possibly cross-organizational business processes.

Recent industry-driven development and standardization efforts have lead to a number of languages for describing service interactions framed within collaborative business processes, most notably WS-BPEL and WS-CDL. At the same time, theoretical research into process-oriented service interaction modeling has been undertaken, leading to various proposals based on well-established formalisms such as Petri nets, process algebra, and communicating state machines. Despite these efforts, it remains unclear which modeling abstractions are most appropriate to capture complex service interactions as found in practical scenarios, and how can these modeling abstractions be combined and incarnated into languages that strike a tradeoff between expressiveness, precision and comprehensibility of the resulting models.

Before attempting to design such modeling abstractions however, it is important to develop an understanding of the requirements that practical scenarios bring to the equation. The work underlying this position paper aims at unveiling such requirements. It sheds light into the nature of service interactions in collaborative business processes, where a number of parties, each with its own internal processes, need to interact with one another according to certain pre-agreed rules. The number of parties in such collaborative processes may be in the order of tens or even hundreds and thus the nature of interactions is rarely only bilateral (between two parties) but rather multilateral. Another crucial feature is that not all service providers in such processes complement each other. Not untypically, they offer similar services and therefore compete. The implication is that canvassed requests to competing service providers require exclusivity – e.g the first response(s) to arrive is (are) accepted while the others are discarded. Also, not all interactions in these environments follow a requestor-respondent-requestor structure. Rather, a sender may re-direct interactions to nominated delegates (or proxies), receivers may outsource requests, choosing to "stay in the loop" and observe parts of responses, and more generally, it may only be possibly to determine the order of interaction at run-time, given, say, the content of messages passed.

The first step of our work has been to capture situations such as those mentioned above as a collection of *service interaction patterns*. Patterns have proved invaluable in the reuse of requirements, design and programming knowledge and expertise. They were traditionally the province of software design, e.g. the well-known (object-oriented) design patterns of Gamma

et al., but have more recently emerged in the BPM field, e.g. through the Workflow Patterns (http://www.workflowpatterns.com). The value of patterns lies in their independence of specific languages or techniques. A pattern, as conventionally specified, captures the essence of a problem, collects references by way of synonyms, provides real examples of the problem, and proposes solutions for implementation in terms of concrete technologies. As such, patterns offer valuable problem-solving insights and allow a particular language's capability to be assessed. In particular, the proposed service interaction patterns have been used to analyze the scope and capabilities of WS-BPEL and to some extent, of related specifications such as WSDL and WS-Addressing.

The collected patterns have been derived and extrapolated from insights into real-scale business-to-business transaction processing, use cases gathered by standardization initiatives (e.g. WS-BPEL and WS-CDL) during their requirements analysis phase, generic scenarios identified in industry standards (e.g. RosettaNet Partner Interface Protocols), and case studies reported in the literature. It is not claimed that the proposed set of patterns is complete. The aim is simply to consolidate recurrent scenarios and abstract them in a way that provides reusable knowledge to service developers and requirements to those designing corresponding modeling languages.

The proposed patterns are structured into four groups derived from the following dimensions:

- The maximum number of parties involved in an exchange, which may be either two (*bilateral interactions*, covering both *one-way* and *two-way interactions*) or unbounded (*multilateral interactions*).
- The maximum number of exchanges between two parties involved in a given interaction, which may be either two (in which case we use the term *single-transmission interactions*) or unbounded (*multi-transmission interactions*).
- In the case of two-way interactions (or aggregations thereof) whether the receiver of the "response" is necessarily the same as the sender of the "request" (*round-trip interactions*) or not (*routed interactions*).

The first group of patterns encompasses single-transmission bilateral interaction patterns. These correspond to elementary interactions where a party sends (receives) a message, and as a result expects a reply (sends a reply). This group covers one-way and round-trip bilateral interactions, but not routed interactions which are covered in a separate group. The second group of patterns stays in the scope of single-transmission non-routed patterns, but deals with multilateral interactions. In this case, a party may send or receive multiple messages but as part of different interaction threads dedicated to different parties. The third group is dedicated to multi-transmission (non-routed) interactions, where a party sends (receives) more than one message to (from) the same logical party. The fourth group is dedicated to routed interactions. Details about the proposed collection of patterns can be found in a working paper available at: http://sky.fit.qut.edu.au/~dumas/ServiceInteractionPatterns.pdf.

The proposed workshop presentation will focus on a subset of these patterns. It is expected that this will trigger a discussion aimed at identifying modeling abstractions that may strike the desired tradeoff between expressiveness, precision, and comprehensibility. This discussion might also shed light into ambiguities and inconsistencies in the patterns and unveil opportunities for extensions and further generalizations.