Journal of Network and Computer Applications 35 (2012) 1304–1316

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Rigorous development of composite grid services

Kenneth J. Turner*, Koon Leai Larry Tan

Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK

ARTICLE INFO

Article history:
Received 16 August 2011
Received in revised form
13 December 2011
Accepted 25 January 2012
Available online 3 February 2012

Keywords:
Bpel (Business Process Execution Logic)
Grid service
Lotos (Language Of Temporal Ordering Specification)
Service composition
Service orchestration
Validation
Verification
Workflow modelling

ABSTRACT

Cress (Communication Representation Employing Systematic Specification) is introduced as notation, a methodology and a toolset for service development. The article focuses on rigorous development of composite grid services, with particular emphasis on the principles behind the methodology. A straightforward graphical notation is used to describe grid services. These are then automatically specified, analysed and implemented. Analysis includes formal verification of desirable service properties, formal validation of test scenarios, testing of implementation functionality, and evaluation of implementation performance. The case study that illustrates the approach is document content analysis to compare two pieces of text. This involves two composite services supported by two partner services. The usability of the service design notation is assessed, and a comparison is made of the approach with similar ones. These show that the Cress approach to developing services is usable and more complete than other comparable approaches.

1. Introduction

1.1. Objectives

The overall goal of this work is an integrated methodology for developing a wide variety of services. A case study is presented for developing composite grid services, but the approach has also been used to create telephony services, interactive voice services, device services and composite web services. The grid (Foster et al., 2002) has emerged as an important approach to distributed computing. Much like the electricity grid, a computing grid provides computational capacity on demand. Grid services can be seen as an extension of web services, with features such as dynamic service selection and distributed resource access.

Communications services are of increasing importance to industry and are therefore becoming quality-critical. The authors believe that practical formal methods such as described in this paper will make a useful contribution to service quality. The methodology described in this article covers service description and specification, verification and validation, implementation, testing and performance evaluation. In industrial practice, service development tends to be pragmatic: services are designed, coded and tested using traditional (and manual) software engineering techniques. However, online services are becoming increasingly mission-critical in many applications. Services are often combined with others for B2B communication (Business-to-Business). The services can be complex, concurrent, and risk unexpected interference. There is a strong need for techniques that deliver dependable services exhibiting desired properties, coupled with automatic code generation to turn these into reality.

Formal approaches are not so common in industry, except in a few specialised application areas such as safety-critical systems. The authors have tried to automate the methodology as far as possible and to make it accessible to non-specialists. Although researchers have aimed at improving service development, they have generally tackled only certain aspects of design and in ways that require detailed technical knowledge.

This article describes Cress (Communication Representation Employing Systematic Specification). Cress offers a notation for services, a methodology for service development, and a comprehensive toolset. Currently Cress handles services in seven different domains, and supports code generation for five different languages. The foundational work in Turner (2000) introduced a notation for telephony features. This was subsequently adapted by Turner (2005) to describe web services and by Tan and Turner (2007) to describe grid services. The service development methodology has recently been rounded out with capabilities for convenient formal verification and implementation evaluation. Relative to previous publications on Cress, this article covers the

* Corresponding author. Tel.: +44 1786 467 423; fax: +44 1786 464 551.
E-mail addresses: kjt@cs.stir.ac.uk (K.J. Turner), larrytkl@gmail.com (K.L.L. Tan).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2012.01.015

complete methodology, grid applications, development principles, and practical formal verification and validation.

1.2. Structure of the article

Section 2 gives the technical background to the work, placing Cress in context. Related work is described on specifying, implementing and testing composite web and grid services.

Section 3 explains service development with Cress. The methodology is illustrated with respect to grid service development, including the subset of Cress used in this article. The techniques for formal verification and formal validation are discussed.

Section 4 describes a case study that makes use of grid services. A document matching service makes use of a scoring service to compare texts for similarity. The diagrammatic service descriptions are automatically specified, verified and validated. Once confidence has been built in the service design, it is automatically implemented and deployed. In a final step, the implementation is automatically evaluated for correct functional behaviour and adequate performance.

Section 5 evaluates usability of the Cress notation and compares Cress to similar approaches. Section 6 rounds off the article with a summary of the approach.

2. Background

2.1. Modelling services with Cress

A service is an abstraction of the functionality provided by an application. SOA (Service Oriented Architecture) has become popular as a means of creating systems from loosely coupled components. A service offers a black-box, interface-oriented view of application functionality. As well as dealing with individual services, SOA also supports composing services into new ones. This offers new business opportunities, but with the added complexity of having to integrate a number of services seamlessly.

Service composition is also called service orchestration or workflow definition. A survey of workflow languages and an assessment of standards in this area are given by Staab et al. (2003). WS-Bpel (Web Services Business Process Execution Logic, Arkin et al., 2007) is a widely used standard for achieving this. Bpel provides the logic that links calls of individual services. Service choreography, e.g. WS-CDL (Web Services Choreography Description Language), is a complementary approach that describes how services interact with each other rather than how their combined execution is achieved.

A composite service is a 'business process' that exchanges messages with partner services. A business process is itself a service with respect to its users. Services have communication ports where operations are invoked. An unsuccessful operation gives rise to a fault that may need compensation to undo prior work.

BPMN (Business Process Modeling Notation, BPMI, 2004) is a graphical notation for describing business processes in general. It is relevant here because it can be mapped to Bpel and thus to web service composition. Compared to Cress, BPMN supports a wider range of capabilities. However, this means that BPMN is a much more complex notation. It also lacks the capabilities of formal analysis conferred by Cress. Furthermore, modelling composite services is only one of many applications of Cress, whereas BPMN is specialised for business processes.

The original grid architecture was OGSI (Open Grid Service Infrastructure), with extensions to WSDL (Web Services Description Language). Grid services now make use of resources through WSRF (Web Services Resource Framework) and also employ GSI (Grid Security Infrastructure). Many researchers have investigated *web* services, but Cress is one of few methodologies that support *grid* services.

Cress respects the principles of SOA and service composition. The emphasis of this article is on formal aspects, here using Lotos (Language Of Temporal Ordering Specification, ISO/IEC, 1989). Lotos is a standardised formal language that was originally designed for specifying networked and distributed systems, but has found application in many other areas. Lotos is a process algebra combined with algebraic data types that supports concurrency, verification, validation and application-defined data types.

Cress is appropriate for domains that can be modelled as a flow of activities; this encompasses a broad range of applications. Cress is not particularly designed for real-time aspects, though there is some support for time and timers. This potential limitation mainly depends on the underlying formalism used. However, the Lotos tools used with Cress support timing and stochastic extensions. Where the implementation language supports time directly, Cress can make use of this. Cress is also not strongly oriented towards performance evaluation. Stochastic aspects (e.g. probabilistic behaviour) are not supported.

Cress supports formal verification (i.e. proof) and formal validation (i.e. rigorous testing). Formal verification requires a finite state space to be practicable, often requiring data type values to be constrained in some way. Formal validation does not have this limitation.

Cress differs from other approaches in a number of respects:

- Most approaches deal with the design of just one kind of service (e.g. voice, web). Cress uses a common set of techniques and tools to support the design of many different kinds of services.
- Most approaches handle only part of service development (e.g. analysis or testing). Cress provides a complete methodology that covers service description, specification, analysis, implementation, testing and performance.
- Other approaches typically need expert knowledge (e.g. formal methods, specialised software). Thanks to a high degree of automation, Cress covers many aspects of development with minimal effort by the service designer.
- Comparable approaches often use non-standard techniques. Cress emphasises the use of standards, with the advantages of wide acceptance, availability of reference and tutorial material, tool support, and attractiveness to industry.
- Although many approaches support the design of *web* services, Cress supports the design of *grid* services as well.
- Formal approaches often abstract services to make them tractable, handling only simple types such as Booleans and integers. Cress supports a full range of data types and data structures that are likely to be required in realistic services.

2.2. Specifying composite services

The Sensoria project (Software Engineering for Service-Oriented Overlay Computers, Sensoria Consortium, 2010) has studied a number of aspects of service design, including larger case studies using web service orchestration. UML (Unified Modeling Notation) is used to describe service structure, evolution and activities. A number of process calculi were created for modelling web services (Wirsing et al., 2006). These are coupled with techniques for functional or performance analysis. UML has also been used in Kaveh and Emmerich (2003) to describe web services at a high level, e.g. using activity diagrams or state diagrams. These are translated into FSP (Finite State Processes) for model checking to