Science of Computer Programming 97 (2015) 202–234

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Rule-based peer-to-peer framework for decentralised real-time service oriented architectures

Alexander Cameron [a,*], Markus Stumptner [a], Nanda Nandagopal [a], Wolfgang Mayer [a], Todd Mansell [b]

[a] University of South Australia, Advanced Computing Centre, Adelaide, SA, Australia
[b] Defence Science and Technology Organisation, Adelaide, SA, Australia

ARTICLE INFO

Article history:
Received 10 July 2013
Received in revised form 8 June 2014
Accepted 8 June 2014
Available online 12 June 2014

Keywords:
Service-Oriented Architectures
Rule based decentralised process execution
Real-time systems
Deterministic
Real-Time Calculus

ABSTRACT

Modularity has been a key issue in the design and development of modern embedded Real-Time Software Systems (RTS) where modularity enables flexibility with respect to changes in platform, environment, and requirements, as well as reuse. In distributed RTS, similar ideas have led to the adoption of Commercial Off-The-Shelf (COTS) components integrated via Service-Oriented Architecture (SOA) principles and technologies that are already well-established in business-oriented information systems. However, current SOA technologies for orchestration, such as Enterprise Service Busses, do not meet strict time-dependent constraints on scalability and latency required by RTS.

We present a novel approach to RTS development where the orchestration of real-time processes is decentralised among the services within a fully distributed rule-driven process framework. Our framework wraps around COTS components implementing individual processing steps in a decentralised real-time process. Our execution model incorporates real-time constraints and is configurable through message routing policies distributed as a knowledge base containing rule sets, and therefore dispenses with the need for a central orchestration component which could easily become a bottleneck. Deterministic behaviour can be achieved through the validation of the rule-sets and the use of Modular Performance Analysis (MPA).

We analyse the performance of our architecture using the Real-Time Calculus and show by empirical evaluation that our method scales to a typical real-time process application found in a tactical naval combat system context.

1. Introduction

Complexity in software systems is a major factor in the decay of software usefulness over time [1,2]. This decay restricts the ability to scale an application or even adapt it to changing business needs. It is a direct product of poor technique and strong coupling between software components and systems that makes them unable to undergo refactoring or modernisation which often results in significant loss of initial investment. The significance of Service-Oriented Architecture (SOA) is

* Corresponding author.
E-mail addresses: alexander.cameron@mymail.unisa.edu.au (A. Cameron), markus.stumptner@unisa.edu.au (M. Stumptner), nanda.nandagopal@unisa.edu.au (N. Nandagopal), wolfgang.mayer@unisa.edu.au (W. Mayer), todd.mansell@dsto.defence.gov.au (T. Mansell).

http://dx.doi.org/10.1016/j.scico.2014.06.005
0167-6423/

that it represents a style where cohesion replaces coupling. That is, where similar components, rather than being coupled together, are organised into layers.

It is these layers that have provided SOA with the ability to absorb new standards and be enhanced by the incorporation of business and infrastructure enhancing components. For example, the Identity Life-Cycle Management System (IDLMS), Enterprise Decision Management (EDM), Service Component Architecture (SCA), Event Stream Processing (ESP), Complex Event Processing (CEP), Enterprise Service Bus (ESB) and, not least, Business Process Management (BPM) are emerging and the most prominent. From our perspective, CEP provides the single most important method of decoupling and as a concept underpins much of our work described in the next sections.

Applying a layered approach to RTS software comes at a cost in terms of performance. Real-time systems are, by definition, time critical reactive systems. They are characterised by being in continuous interaction with sensors or other supporting systems and producing responses within a specified interval of time. Examples of RTS can be found in modern motor vehicles, ship-based combat systems, and aviation systems. As they are time critical, they are often embedded, highly efficient and designed for a specific purpose. In this regard they are less flexible than systems developed using SOA, in general and, as a result, naturally age with the system they were designed to support.

Real-time systems are generally classified as being "*Hard*" when the system must deterministically meet all deadlines. In order to achieve this, the only feasible approach has been to design dedicated hardware and software components that are tightly integrated in a time-synchronous manner. Although strong guarantees can be proved for such architectures, they also come at a high cost, and may, as already stated, have limited flexibility and adaptability. On the other hand, "*Soft*" or "*Firm*" means that a degradation of system performance arising from occasional failures to meet a real-time constraint can be tolerated. In a typical mission critical system, this tolerated rate of failure is one failure in 10^5.

The question that therefore arises is whether real-time systems in the military can benefit from the advances that networked and enhanced SOA offer. Also, can issues such as meeting real-time constraints and the analysis of distributed, asynchronous systems involving "black-box" components allow the derivation of formal guarantees concerning deadlines and the achievement of Verification and Validation formalisms?

2. Motivation

Networked Service-Oriented Architecture is a distributed architectural style that has evolved to address the need for scalable interoperability through realising the full value of available components. Within the military, networked SOA has been seen as vital for some time and to that end has penetrated the operational or organisational level and is gaining the ascendency over previous approaches to enterprise software development. However, the application of standard SOA design principles within the often asymmetric tactical domain remains a challenge. Nonetheless, it is seen as the next step in the evolution of mission critical Command and Control (C2) systems [3] and this is behind the activity to introduce new standards and adapt existing ones. The primary concern is achieving a balance between the importance of standards such as HTTP and SOAP for service activation and the overhead caused by use of verbose XML [4].

The challenge for distributed SOA real-time systems is to achieve the desired modularity whilst remaining true to the basic tenets of SOA. For example, open standards for service invocation, asynchronicity and loose coupling all pose serious obstacles. Apart from these, services of varying granularity and centralised orchestration can also lead to non-optimal processes and make deadlines difficult to achieve. Although at first glance varying granularity may not appear to be an issue, it does play an important role in achieving higher levels of reuse and reduced development and maintenance costs [5]. It is these two factors that contribute most to the decay of software. Finally, centralised orchestration, apart from not being scalable due to the need to transfer data from the services to the central orchestration engine, also suffers from being a single point of failure. Thus, in many respects we see a basic mismatch in technology between what needs to be achieved with RTS and what is offered by SOA.

We believe there is a need for hybrid architectures, where there is a natural decoupling of components of the system that need to be hard real-time and time synchronised, such as in the case of sensors and effectors with those components that can be made more interoperable by operating in a more discrete and asynchronous manner. The motivation for such hybrid architectures comes from the desire for military systems to be able to distribute and re-combine in new configurations to share data and information in order to achieve an information advantage over platform-based adversaries.

2.1. Related work

SOA implementations in the business domain typically incorporate an "Enterprise Service Bus" (ESB) or "Message Broker" which orchestrates the overall service process execution by mediating between individual services and routing messages. Common SOA implementations combine an ESB and a service based business process orchestration engine such that the individual services are executed under the control of the process engine which interprets a business process model, such as the Business Process Execution Language (BPEL) or Business Process Modelling Notation (BPMN). The prevailing implementation rests on a centralised process engine for orchestration [6]. Although distributed and fragmented implementations of ESBs exist, the overall SOA performance often falls short of what is required to reliably meet RTS guarantees. Overheads incurred by standards based service invocation protocols and latencies arising from limited network capacity are common causes which exacerbate poor performance [3].