LogOS: an Automatic Logging Framework for Service-Oriented Architectures

Stéphane Frénot
Université de Lyon
INSA-Lyon, CITI-INRIA
F-69621, Villeurbanne, France
stephane.frenot@insa-lyon.fr

Julien Ponge
Université de Lyon
INSA-Lyon, CITI-INRIA
F-69621, Villeurbanne, France
julien.ponge@insa-lyon.fr

***Abstract*—As multi-source, component based platforms are becoming widespread both for constrained devices and cloud computing, the need for automatic logging framework is increasing. Indeed, components from untrusted and possibly competing vendors are being deployed to the same runtime environments. They are also being integrated, with some components from a vendor being exposed as a service to another one. This paper presents our investigations on an automated log-based architecture called LogOS, focused on service interactions monitoring. We developed it on top of Java / OSGi to enable identification between bundle providers in cases of failures. We motivate the need for an automatic logging framework in service-oriented architectures, and discuss the requirements of such frameworks design. We present our implementation on OSGi. Finally, we position our approach and give some perspectives.**

***Keywords*-soa; cbse; osgi;logging; service;**

I. Introduction

While software developers and integrators have been using logging systems for technical purposes, we argue that a new breed of need of logging approaches is emerging from an

architecture ported to Java / OSGi, and how the aforementioned requirements need to be implemented for this platform. Finally, we present our experiments with our OSGi-based implementation, before concluding with related work and perspectives.

II. Designing an Automatic Log Record Management Framework

Our proposed architecture focuses on service oriented logging for component based architecture. It focuses on two principles for component activity logging: horizontal calls and black-box approaches. In horizontal calls, every available service function is provided by another component, either remotely or locally, and invocations are sent to the same layer. There are no existing trusted relationships between callers and callees, in the sense that callers and callees have the same level of legitimacy within the system. Figure 1 illustrates, horizontal