

Customizing the behavior of middleware: the PIE approach

G. Cugola¹, P.Y. Cunin², S. Dami², J. Estublier²,
A. Fuggetta¹, F. Pacull³, M. Rivière³, H. Verjus^{2,4}

¹ Politecnico di Milano, Italy.
Cugola@elet.polimi.it, Alfonso.Fuggetta@polimi.it

² LSR laboratory, Grenoble University, France.
{Pierre-Yves.Cunin, Samir.Dami, Jacky.Estublier}@imag.fr

³ Xerox Research Centre Europe, Grenoble France.
{Francois.Pacull, Michel.Riviere}@xrce.xerox.com

⁴ LLP/CESALP laboratory, Savoie University, Annecy France.
verjus@esia.univ-savoie.fr

1 Introduction

Started in February 1999, PIE¹ is a two-years LTR project founded by the European Union. PIE's main goal is the development of a platform for "Process Instance Evolution" in a federated Process Support System (PSS).² With the term "federated PSS" we refer to a federation of distributed Commercial Off The Shelf (COTS) tools that provide support and automation to human-intensive activities. To pursue this goal, the PIE federation relies upon a new generation middleware, capable of offering an advanced set of services to support dynamic reconfiguration and customization of the federation's operations. We called this middleware the "PIE middleware". The paper briefly outlines the main features of the PIE middleware. In particular, we concentrate on the services offered by the middleware to change its own behavior and operations.

2 Features of PIE middleware

A preliminary analysis of the requirements of the PIE federation convinced us that most of the properties we require for federation control could be translated into communication control. This means that the PIE middleware has to offer a set of services to control the communication among interacting components. At the same time we observed that to cope with changes in the way components interoperate and to support the introduction of new communication styles and policies, the PIE middleware has to offer the ability of adding new services and, therefore, changing its own behavior. As a consequence, the PIE middleware adopts a sort of *plug-in approach* that makes it possible to incrementally expand and enhance the middleware own features.

The set of services offered by the PIE middleware can be roughly classified as follows:

- *Basic communication services.* The PIE middleware has to offer both message-based and service-oriented services. To enhance interoperability, these services have been designed to comply with two emerging standards, namely, the Java Messaging Service and Java RMI.
- *Services to change the middleware behavior dynamically.* This feature is supported through a plug-in approach. PIE plug-ins are called *handlers*. A handler is a piece of code that is invoked by the PIE middleware upon specific conditions. Its purpose is to change or extend the policies used by the PIE middleware to manage a communication request. The PIE middleware provides a set of services to make it possible to add and remove handlers.

By exploiting handlers it is possible to change the policies used for message routing, change the content of a message, and implement new, advanced services (see next point).

¹ PIE: Process Instance Evolution. Esprit Project 34840. UJF Grenoble, Victoria U. Manchester, Dortmund U., Savoie U., Politecnico di Milano, Xerox Grenoble, Teamware, Dassault Systèmes.

² Another term that is often used to refer to a PSS is Workflow Management System.

- *Enhanced communication services.* An important application of handlers is concerned with implementing new communication services. For instance, by using handlers it is possible to support group and transactional delivery of a set of related messages.
- *Services to support mobility.* To evolve a large, distributed system, it is necessary to support dynamic change in the architecture of the system by allowing components to move from a host to host. The PIE middleware offer a set of services to allow PIE components (COTS in the PSS federation) to disconnect from the middleware infrastructure and reconnect from a different location. Once reconnected, a PIE component is able to retrieve the messages and communication requests that have sent to it while it was disconnected.

3 The architecture of PIE middleware

The left side of Figure 1 shows the logical structure of the PIE middleware. It is composed of a *core* (a set of *handlers*), and a set of *shared data structures* (i.e., shared among different handlers). PIE clients interact with the PIE middleware through a library, which implements the PIE middleware API. The role of the PIE middleware core is to manage handlers activation.

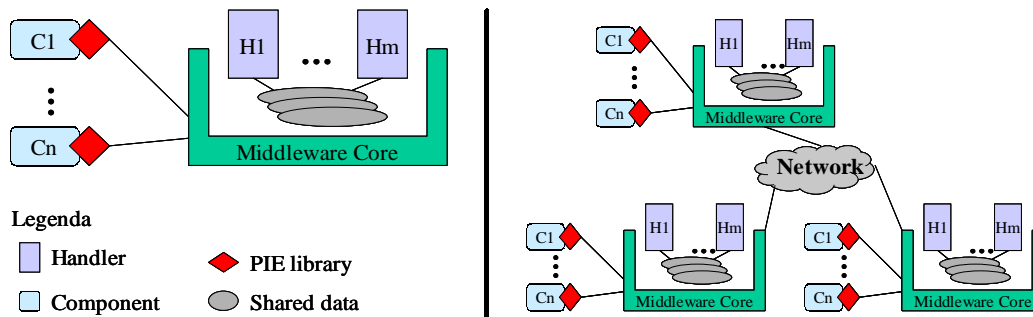


Figure 1 The logical and physical architecture of the PIE middleware

To increase the scalability of the PIE federation both in terms of the number of possible components that interact through the middleware and in terms of the “network distance” among these components, the PIE middleware adopts a distributed architecture in which a number of distributed dispatchers cooperate to provide the services outlined in Section 2. Each dispatcher has an internal architecture equivalent to the one described above. It is composed of a core and a set of handlers and it is connected to the other dispatchers through network links. Right side of Figure 1 gives an overview of this architecture.

4 Conclusions

Supporting changes in a federated environment is a complex task that requires advanced middleware services. As part of the PIE project, we are developing a new generation middleware capable of supporting such task and able to evolve through the adoption of a plug-in approach. The PIE middleware is being tested within the framework of a research project aiming at developing a distributed infrastructure for federated process support.