Costs and benefits of Pilarcos federation middleware

Lea Kutvonen

Department of Computer Science, University of Helsinki

P.O. Box 26 (Teollisuuskatu 23), FIN-00014 UNIVERSITY OF HELSINKI

February 10, 2003

Abstract

Current information processing needs of companies require inter-organisational cooperation. New middleware is expected to provide facilities for capturing the workflow, searching for available members for the workflow, and ensuring that a suggested set of members is able to interoperate both in semantical terms and technically. The Pilarcos project has developed a prototype middleware where inter-organisational application management is based on explicitly stated, platform-independent, multilateral contracts that define the forms of cooperation between performing components. This paper discusses the performance costs caused by the middleware and the benefits gained by the use of such middleware. As a conclusion the applicability and requirements of the adoption of Pilarcos middleware are discussed.

1 Introduction

Several phases of enterprise application integration technologies (EAI) have emerged in the recent years and improved the availability of IT services of large companies. Internationally, it can be seen that for the next few year, the research and development trend will turn to agentified, dynamically collaborating business systems. Because these integration solutions are founded on a single authority directing business models and their technical development, the approach is not applicable for small and medium scale companies (SMEs). The small and medium scale companies by necessity focus on a small set of novel IT services, and provide them on a relatively small, seldom changing set of IT platforms (communication and information processing systems).

The SMEs gain and keep their competitive position best by networking with companies that provide complementary services.

Networking of enterprises requires support for dynamically creating cooperation relationship, federations, between services provided by each enterprise. There is need for describing new kinds of federation structures, joining and leaving federations, and changing membership of federations as the market situations change.

Federation management is made problematic by two major issues. First, at each enterprise different technology solutions for information processing and communication used. Second, each enterprise has its private set of policies governing its business decisions and operational solutions in practise. Federations cannot be allowed to override neither of these decisions but must incorporate negotiation of choices and a set of transformation elements for overcoming technical differences.

A federation management infrastructure for enterprises essentially provides facilities for composing new services together with other enterprises and, in order to do this in a reasonably large scale, mediation between platform technologies. This structure supports autonomous evolution of the IT system in each enterprise and also masks changes in the cooperating systems.

The Pilarcos project develops infrastructure services for federation support in such a way that inter-organisational cooperation can be defined by a workflow-like, dynamically changeable description. For the federations, participants can be selected dynamically and and the negotiated cooperation policies effects the final workflow structure. Management of the federation is distributed and each cooperating party is autonomous.

The Pilarcos project has developed a prototype middleware where inter-organisational application management is based on explicitly stated, platform-independent, multilateral contracts that define the forms of cooperation between performing components.

This paper briefly introduces the middleware, discusses the performance costs caused by the middleware and the benefits gained by the use of such middleware. As a conclusion the applicability and requirements of the adoption of Pilarcos middleware are discussed.

2 Pilarcos middleware services

Pilarcos middleware provides application programmers with pervasive, platform-independent tools to manage federations. However, the use of the functions remains explicit. In this section we review the concepts provided for the programmers and the middleware functions available for building applica-

tions [9].

The key concept for inter-organisational cooperation is that of federation. Federation is an identified and structured collaboration of a group of partner components in peer-to-peer communication relationships. Each federation is maintained by a federation contract. A *federation contract* captures the structure of the federation by reference to a business architecture, the selected members of the federation or selection rules for membership, and policy decisions agreed for the particular federation.

A *business architecture* is defined by a set of roles, interactions between roles and a set of policies. The business architecture description does not fix the identities of the participating systems. Instead, roles are associated with *service type* that defines the the class of service required. Members for the federation are selected based on service offers exported to trading services. The service offers include meta-information on component's service type, technology requirements, conversation protocols expected, operational policies, cost, location, etc.

Policies in a business architecture have two targets. First, a policy rule can be set to govern the behaviour of a component in a role. For example, different information retrieval strategies can be preferred depending whether there is need to save space or time in a search. This kind of expectation can be passed on to a component via a role related policy. Second, a policy rule can be set to govern interactions in the federation. The business architecture can model alternative interaction sequences and the policy value can be used to determine which of the alternatives should be used for the federation.

Component programmers provided with a repository from where business architectures can be retrieved at need. There are separate processes for designing and publishing business architectures for a different group of practitioners. The business architectures are considered to be globally available and interpretable via federated repositories. Service types related to roles can be interpreted also as requirement definition – besides selection criteria for components.

In the Pilarcos project, no new software production tools are provided. However, we see the recent development of OMG MDA (model driven architecture) [8] tool chain a complementing approach. Here, a component is to be understood loosely a service implementation encapsulated in such a way that platform services are able to manage its life-cycle (deployment, instantiation, termination, activation and deactivation). Although Pilarcos project has used component based platforms (OpenCCM, MicoCCM, JBoss) for experimenting and prototyping, the Pilarcos architecture does not require component techniques to be used.

Requirements for components cover two aspects. First, the components

are expected incorporate Pilarcos federation management interface. This interface defines operations for policy manipulation, request for federation establishment or termination. Second, the components are expected to follow policy rules stated in the local policy repository.

The Pilarcos management services enhanced trading, federation management, type management, federated binding, policy repositories follow RM-ODP model of middleware. The RM-ODP standard by ISO and ITU defines – in addition to terminology and viewpoints – a middleware model ([3], clauses on engineering viewpoint, functions and transparencies) that can be used to support inter-organisational applications.

The enhanced Pilarcos trader provides two main operations: exporting a *service offer* for a specific service type (`export`), and populating a business architecture with mutually compatible service offers (`populate`). The former operation is used by an administrative tool used by a service provider. The latter operation is used by the Pilarcos federation manager on behalf of the application wanting to establish a federation. The `populate` operation takes an incomplete federation offer as a parameter, and returns one or more completed federation offers. No separate constraint parameter is used; instead, the incomplete federation offer typically contains a *pre-filled* service offer for the populating role itself, defining its policies for the federation. Thus, the population process is completely symmetrical: any role that has been left empty in the incomplete federation offer is populated by the Pilarcos trader. This makes it easy to do partial re-populations for failure recovery or adaptation purposes.

The federation managers are responsible of running the protocol for negotiating, maintaining and re-negotiating federation contracts. For federation managers, the essential information element is of type *federation offer*. It is a combination of compatible service offers, one for each role in a specific business architecture. The federation establishment protocol is initiated by a client request. As a first step, a service offer that describes the client itself is positioned into a federation offer element. The Pilarcos trader then populates the rest of the roles. As a result, several suggested federation offers are returned for the client to choose from.

For testing whether a component is suitable for a federation, service type matching is needed. The Pilarcos middleware design includes an enhanced version of ODP type repository [2] for holding relationship information between generic types (service types, binding types, interface types) that are technology-independent and used for matching purposes and technology-dependent templates that are used for instantiating the corresponding components and objects. This mapping information is created by system programmers separately from business architecture descriptions and service of-

fers.

Pilarcos middleware expects that components can be managed by policies, much like in policy-based management systems (e.g. [7]). Administrators can create and set policies for component groups. Furthermore, federation contracts are stored into the policy repository and thus federation contracts become an integral part of component management mechanisms. The current implementation is bare.

Federation contracts are formed using platform-independent models. In order to realise federation management events, the abstract notations must supported with mappings onto technical realisations and service instances. For example, platform-independent requests for deployment, instantiation or binding need to be mapped on platform-specific installation scripts and factory services. Especially, federated binding requires both contractual and actual modes and transitions between the modes, as described in ODP binding framework [4].

The Pilarcos approach uses two types of domains: administrative domains and technology domains. An administrative domain can be for example an organisation, a company or a department with authority to do independent operational decisions about the way it runs its business. Organisation-wide policy management is needed to allow IT-system administrators to reflect operational policies – such as restrictions to cooperation partners, payment related conversation styles and time of availability of offered services – consistently onto all applications of the organisation in an automated manner. Service descriptors, service management rules, and policies are defined at the administrative domain level in technology-independent terms.

Component management rise the need of separation between technology domains. For each technology, there are various facilities for deployment, instantiation and termination of components. A technology domain is here limited within an administrative domain for simplicity. At each technology domain, service descriptors and service management rules are mapped onto technical engineering solutions. Naturally, these mappings follow a pattern common to all administrative domains.

The Pilarcos middleware services reflect the administrative and technology domains and the need to cooperate across the domain boundaries [9]. There are collaborative middleware services with a running agent at each administrative domain for negotiating federations and advertising available services. These agents take care of making requests to their peers at other domains, as there is no authority to otherwise invoke management actions at a foreign domain [5]. The requests carry contracts to pass relevant meta-information that identifies what should be done and how. On the other hand, there are local management facilities running at each technology domain for

managing components. In addition, at each administrative domain there are agents responsible of mapping abstract federation contract information into a form usable at technology domains.

3 Costs of Pilarcos middleware

While Pilarcos middleware provides facilities for dynamically forming federations there is overhead cost from the use of the middleware services. This cost has been estimated by a series of performance measurements on the prototype application and infrastructure services. First, the overall overhead of Pilarcos middleware in terms of CPU load and effect on response times seen by a client was considered. Second, the main elements of the cost were factored out at the areas of business architecture population, federation establishment and termination and adaptation of communication across CCM and EJB platforms. Third, some scalability tests were run to see how input rate of client requests affect the response times.

The measurements were performed with a system that included all the Pilarcos infrastructure services and an application case. In concrete terms, the Pilarcos prototype consists of Pilarcos infrastructure service components and Tourist Information Service application components that take advantage of the Pilarcos services. Infrastructure services and most of the application components have been implemented on two new CORBA Component Model platforms, the Java-based OpenCCM and the C++-based MicoCCM [**?**]. In addition, one of the application domains is built on Enterprise Java Beans technology, running on the JBoss application server. Seamless interoperability between the three platforms has been a major focus the prototype development.

Figure 1 illustrates the main Pilarcos infrastructure components and their relations in a generic setting. The organisational domain borders separate the organisations running the client application, the server application, and the global trading and type repository services

The application case is built around an idea of a portal service, the *Tourist Info*, which provides travellers access to vertical tourist services like travel information, hotel bookings, and weather services. It is assumed that neither the portal service nor the vertical services are free, and the traveller has some electronic payment instrument (e.g. credit card) available.

The prototype implements two business communities: tourist info community and hotel info community. Tourist info community contains three domains each with their own role (tourist info client, tourist info service, payment service) and describes the business community related to providing

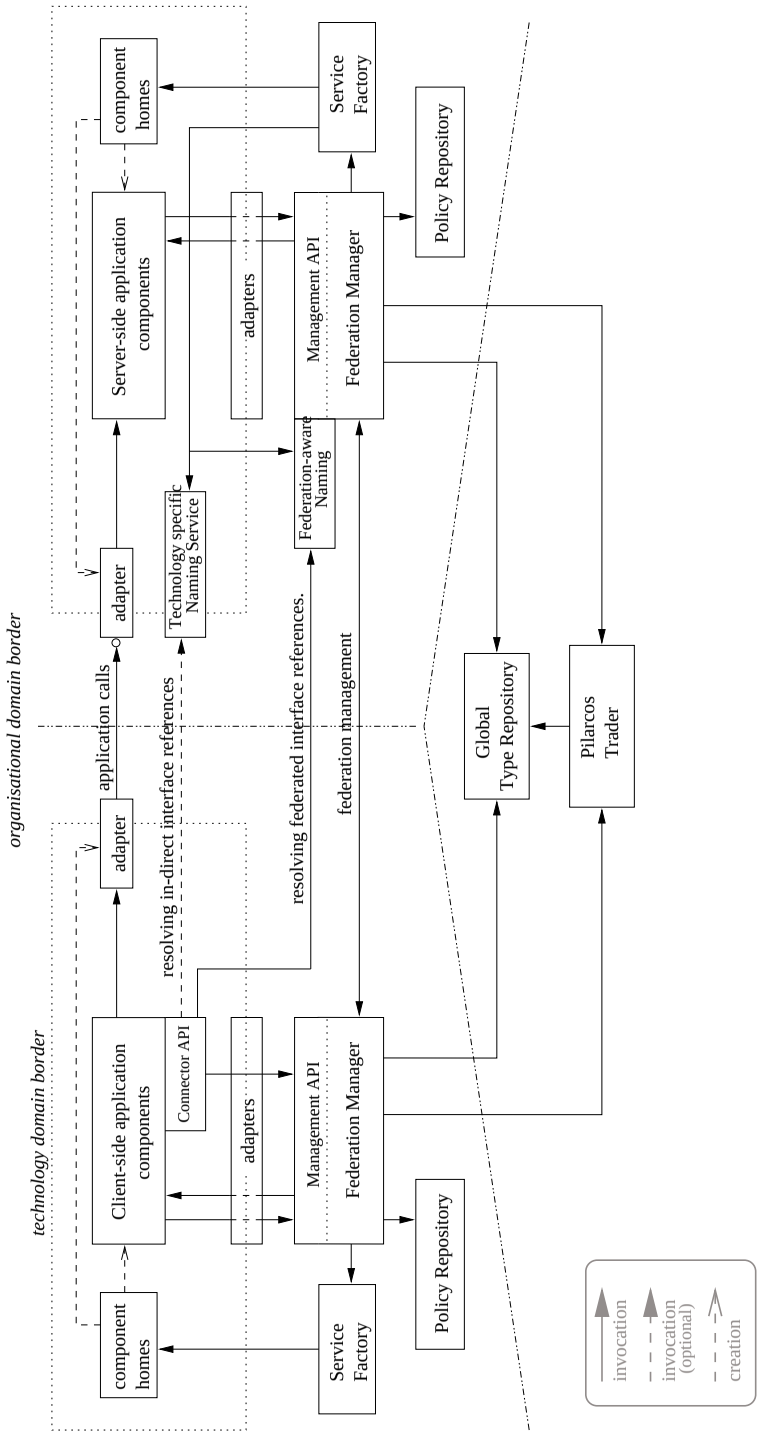

Figure 1: Overview of Pilarcos prototype components[9].

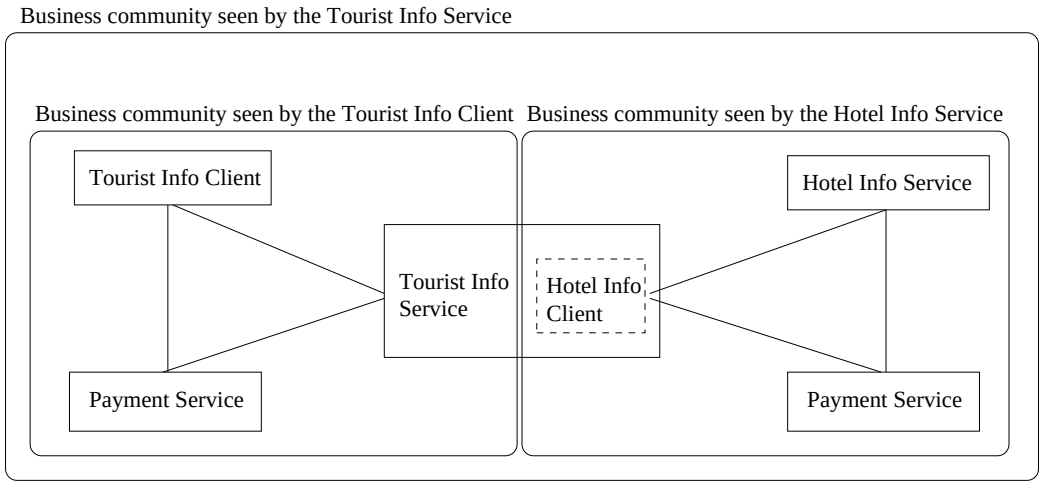

Figure 2: Business entities and communities in the *Tourist Info Service* case[9].

the portal service. Hotel info community also contains three domains each with their own role (hotel info client, hotel info service, payment service) and describes the business community related to providing the only implemented vertical service (hotel bookings). Tourist info client and hotel info client represent the users of corresponding services whereas tourist info service and hotel info service represent the providers of the services. Payment service represents a trusted third party used to mediate the transfer of funds between the other entities in the community. Figure 2 shows the business entities and the communities formed by them.

Measurement environment consists of seven 1GHz Pentium III workstations with 512MB RAM memory. Workstations were connected with closed 100 Mbps Ethernet. Operating system used in workstations was CS Linux with kernel 2.4.18. Different components of Pilarcos prototype were distributed to workstations, one per each machine. Background clients used two machines and were evenly distributed in them. Distribution of components are shown in Figure 3.

Before measurements the system was warmed up with 3000 measurement rounds done by client. The measurements themselves included 3000 rounds by one to 8 clients. Throughput optimisation flags were used for Java virtual machines. In all measurements IBM Java 1.4.1-platform was used.

Looking at the measurement results, the cost of Pilarcos middleware usage appears to be acceptable. As we consider a "session" form a user, the application runs the following steps:

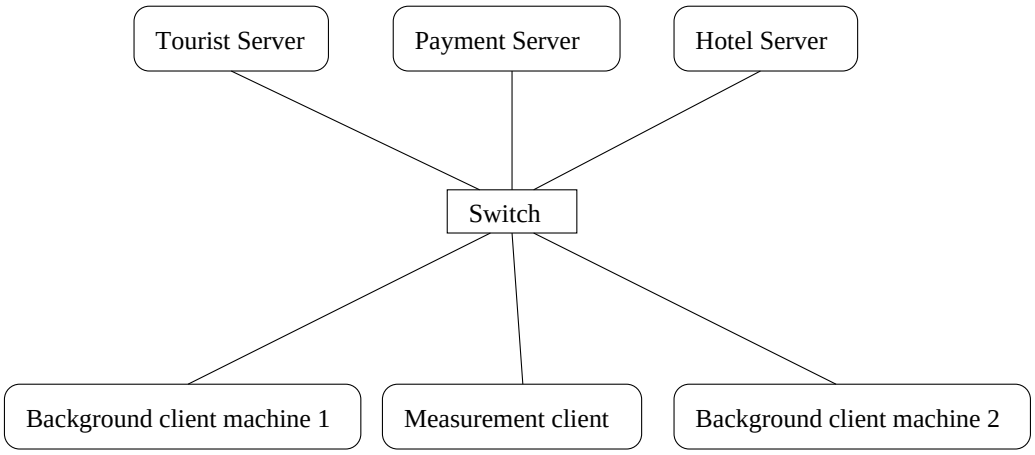

Figure 3: Distribution of Pilarcos components in measurement environment[9].

- ask the Pilarcos trader to populate a business architecture;
- tell the federation manager to create the federation; this actually involves the population and creation of the second federation in the application scenario;
- run a group of application scenario specific operations and requests for the servers;
- end the session and ask the federation manager to terminate the federations.

The population, federation creation, and federation termination phases can be seen in Figure 4. Under normal load conditions, the population process takes less than 25 milliseconds, federation creation under 40 milliseconds (the measured 100 ms includes two create operations and a population operation), and federation termination about 20 milliseconds (again, the measured numbers cover two terminate operations). Operations for getInformation and makeReservation are also noticeable in the measurements. The 20 milliseconds of getInformation include adaptation delay between the platforms, the 30 milliseconds of makeReservation additional factor of bean instantiation on the EJB platform.

For the measurements presented in Figure 4 the number of request making client computers were increased so that the input rate of requests reached a level where saturation on one of the computers was approaching. However, no unexpected behaviour on response times is seen here.

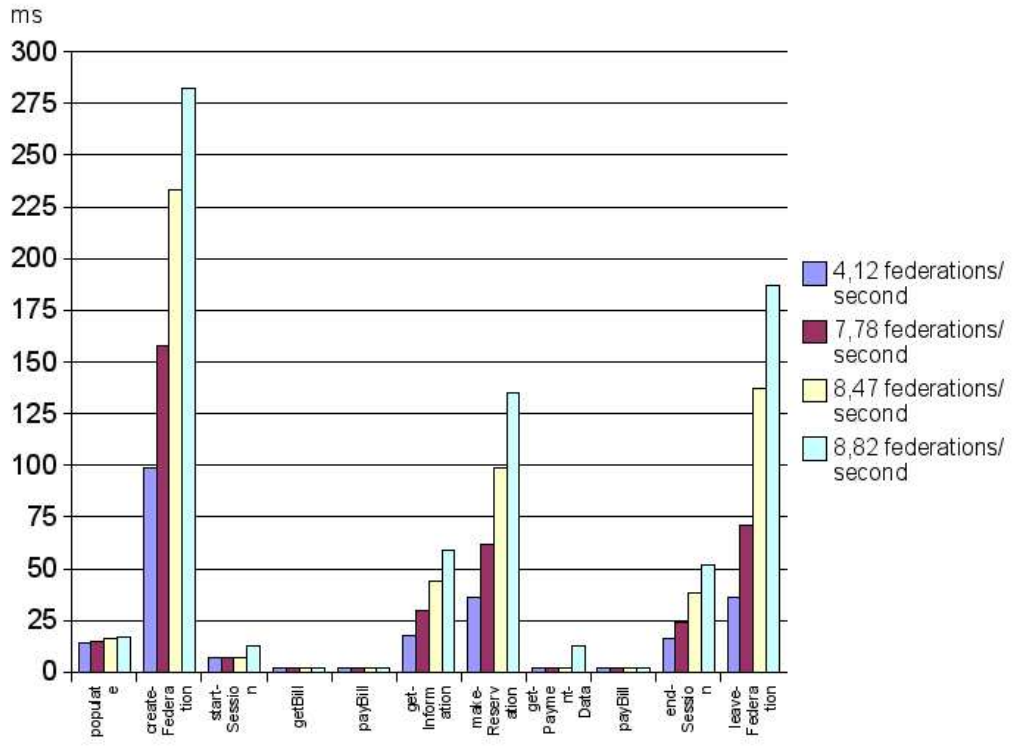

Figure 4: Response times for the client[9].

Looking at the CPU usage, additional measurement needs arise. The measurement environment was initially set up in such a way that one of the computers run two platforms simultaneously. This computer runs the Hotel Server application component and takes care of all platform specific transformations and adaptation needs. Figure 5 shows that this computer is becoming a bottleneck in the system as the load increases.

The conclusions from this behaviour are twofold. First, additional measurements are needed in an environment where the platforms are (more realistically) installed in separate computers. Second, the cost of adapter creation is considerably high in the current implementation. A library based solution has been planned but was not yet available for measurements.

To support interpretation of the above measurement results, Figure 6 shows the same application components running without the help of Pilarcos middleware. The configuration of application components is here fixed, and only OpenCCM is used as a platform. Thus, the flexibility of finding appropriate partners for the federation is missed, and also heterogeneity support. All application components have been selected beforehand and their locations

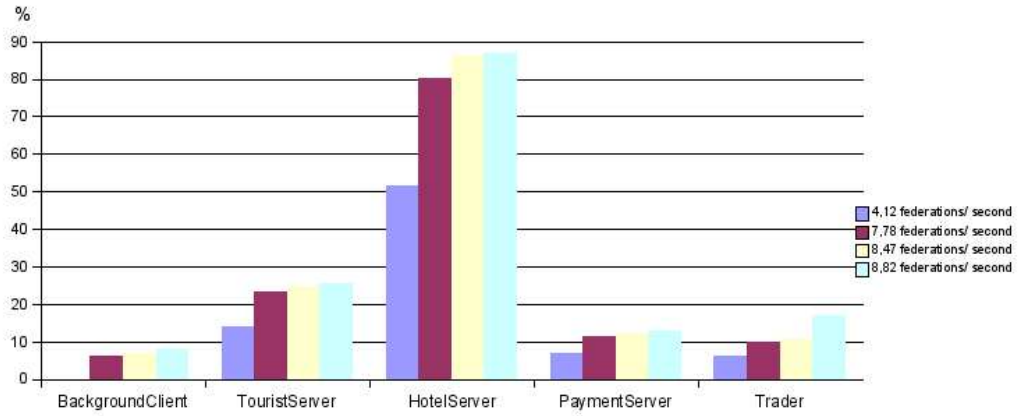

Figure 5: Processor usage in servers.

are resolved by name server. The measurements show that the overhead load is mostly visible in the new operations offered by the Pilarcos middleware. Additional few milliseconds fixed cost is added on all operations.

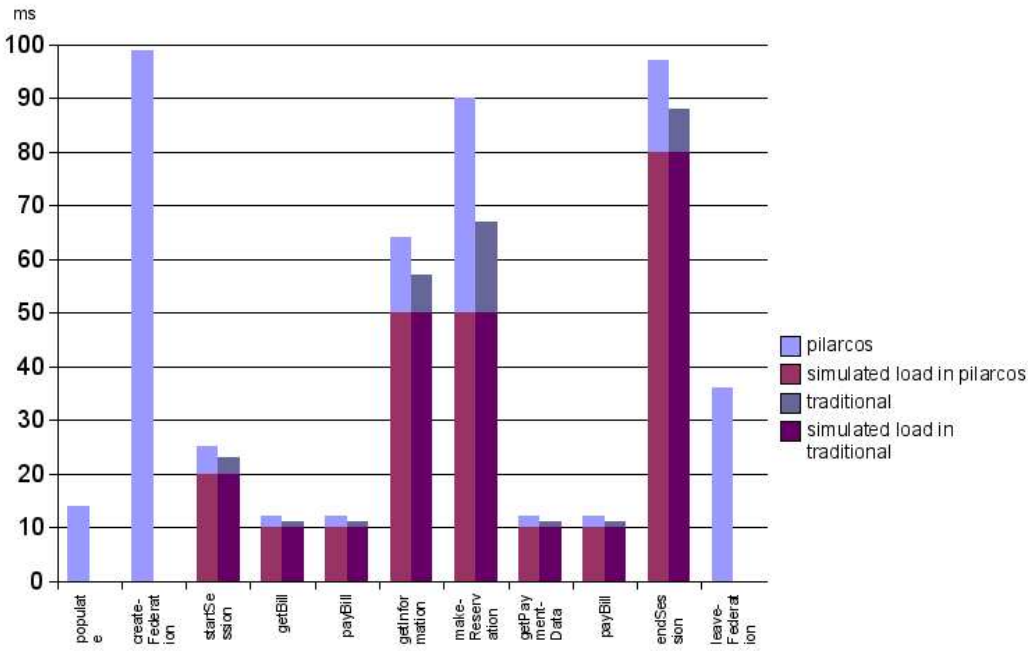

Figure 6: Time used in different phases seen by client with simulated load[9].

In general, the measurement results were as expected. However, the use of

two platforms in one computer is unrealistic and further measurements should be done. Second, platforms are not mature enough and caused scalability problems themselves, so scalability tests were not done as thoroughly as we wished.

4 Benefits of Pilarcos middleware

This section discusses the ways in which Pilarcos middleware extends an organisations possibilities to establish cooperation relationships and still preserve its autonomy, and how the new middleware services effect on software development, maintenance and administration.

4.1 Forming cooperation relationships

The Pilarcos middleware makes it possible to form application level cooperation relationships between multiple equal partners. Federations can be formed across organisational boundaries and over technology domains. In simple cases new middleware services just make programming easier, but more complex cooperations could not be achieved otherwise.

In simple cases, the Pilarcos middleware helps in reducing application software complexity by providing ready-made functions for service discovery and binding in heterogeneous platform environments. Currently, most service-oriented architectures built with Web services middleware, in Java environments or with CORBA support (plain or with components) concentrate on service discovery in client-server type of situations. Services are considered as a set of independent resources in the global network, as building blocks, from which an application programmer is responsible to construct a new value-added service. Most middleware solutions solve the interoperability problems by either trusting a shared language context (Java Virtual Machine) and transfer of code, or trusting a shared protocol environment (CORBA, Web services). In Pilarcos, interoperability tests use explicit information on the shared platform facilities, thus making it possible to use different technologies side-by-side as long individual federations find common communication facilities.

In more complex cooperation networks it is clear that without Pilarcos middleware the application components could not establish federations, i.e. multilateral contracts, in such a flexible way. Inter-organisational cooperation can only be achieved through standard solutions and in this case, via standardising new middleware services and meta-information elements.

4.2 Extending the potential cooperation networks

The Pilarcos approach is not dependent on one communication standard; in contrary, the goal is to govern the multitude of communication standards. In practise this means that service discovery and binding mechanisms carry explicit information on the protocols involved and that there is a set of adapters available for the most common combinations.

For organisations this means expansion of potential cooperation partners from those who use the same kind of communication platform (for example, an ORB from the same vendor). The cost of adapted communication is higher, but the amount of manual administration and the number of platforms purchased is kept in minimum.

4.3 Preserving autonomy in cooperations

In contrast to distributed workflow systems, the Pilarcos architecture design deliberately denies access and visibility to partner's information processing system. A business architecture can be considered as an external workflow template that describes potential choreographies between independent parties. Each potential party provides a local workflow that fulfils its published service contract. The local workflows are hidden and no specific requirements for their running environment is specified. The Pilarcos middleware acts as a matchmaker for the parties, and then specialises the shared external workflow according to the membership, corresponding business policies and technical facilities.

The Pilarcos architecture allows organisations to simultaneously participate several cooperation relationships that have different requirements on communication and policies. Each of the federation contracts must be compatible with the organisation's own policies and facilities, but within those limits, the federation contracts participated can even be contradictory to each other.

The separation of local and global workflows supports independent evolution of services in each organisation. It is possible to change local implementations or platforms without cooperation partners to notice the change. This allows organisations more time to react on external pressures. Especially for SMEs this is beneficial. The SMEs gain and keep their competitive position best by networking with companies that provide complementary services. Current integration solutions are founded on a single authority directing business models and their technical development, thus benefiting only very large companies.

4.4 Separation of concerns on development

Separating design of business architecture descriptions, design of service types, and implementation of components improves the software engineering process in general. The natural lifetimes of these elements differ, as well as the skills required for the provision of them.

The Pilarcos middleware services allow workers to concentrate on their component logic and provides business architectures and technology mappings as ready-made solutions, as instructed by the currently popular production line concept [1].

As the Pilarcos middleware is based on repositories, like type repository and trader, that can federate with each other, it is possible to make the software development process a distributed one. The pieces for a federation, in this case a pre-designed distributed application, can be implemented by independent teams and existing components can be reused where applicable.

The Pilarcos design supports component-based service markets by providing a mechanism for explicating the services needed. The role definitions in business architecture descriptions give precise description of the services needed and the functionality of the component packages that should be easy to market. As a sign of development in this direction can be seen in OMG domain services, where new services need to define the intended use in addition to their interfaces.

The Pilarcos design supports also evolution of business architectures. As business architectures are explicitly defined and published, differences in versions can be detected and problems caused by differences in process assumptions can be avoided. The publication mechanism also supports development and delivery of new versions of business architectures. With epoch related enhancements in the business architecture description language, even existing federations can be made to transform their cooperation to use a new version of a business architecture.

4.5 Reduced cost on administration and maintenance

Providing tools for automating administration of service composition reduces system maintenance cost. Because the federations directly support changes in membership, there is good support for adaptation to changes in business situation and also to changes in technical availability of services. The design incorporates into the same management model both private business within the organisation and external cooperations with varied and potentially contradictory requirements.

The federation mechanism helps in the maintenance of the application:

the overall architecture can be changed by changing the business architecture; the components can be updated by publishing new versions or compatible services; and new platform technologies can be adopted by creating and publishing a set of adapters.

5 Conclusions

Adopting Pilarcos middleware requires that standard, pervasive management services were adopted generally. These standards should cover the cooperative services between middleware platforms, especially protocols for federation managers. Also, a language or appropriate language sets should be standardised for expressing the meta-information models for cooperative management across organisations. The information models required include business architecture models, federation contracts, service offer contents, and type definitions [6].

The Pilarcos middleware is applicable in situations where flexibility and evolution support is required. Especially, for small and medium size enterprises the networking support the middleware provides may give an essential benefit. However, the open nature of Pilarcos model is such that highly secure systems cannot be supported: trust building mechanisms can be incorporated, but still, critical systems should use more static and closed solutions. The nature of Pilarcos middleware overhead cost is such that it is not suitable for example for hard real-time systems or systems where response times are not allowed to vary a lot.

Acknowledgements

This article is based on work performed in the Pilarcos project at the Department of Computer Science at the University of Helsinki. The Pilarcos project is funded by the National Technology Agency TEKES in Finland, together with Nokia, SysOpen and Tellabs.

References

[1] Herzum, P., and Simms, O. *Business Component Factory.* Wiley Computer Publishing, 1999.

[2] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. ODP Type Repository Function.* IS14746.

[3] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 3: Architecture*, 1996. IS10746-3.

[4] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing – ODP Interface References and Binding*, Jan. 1998. IS14753.

[5] KUTVONEN, L. Management of Application Federations. In *International IFIP Working Conference on Distributed Applications and Interoperable Systems (DAIS'97)* (Cottbus, Germany, Sept. 1997), H. Konig, K. Geihs, and T. Preuss, Eds., Chapmann & Hall, pp. 33 – 46.

[6] KUTVONEN, L. Automated management of interorganisational applciations. In *EDOC2002* (2002).

[7] LUPU, E., AND SLOMAN, M. A policy based role object model. In *Proceedings of the 1st International Enterprise Distributed Object Computing Conference (EDOC'97), Gold Coast, Queensland, Australia* (October 1997), pp. 36–47.

[8] SIEGEL, J. *Developing in OMG's Model-Driven Architecture.* Object Management Group, Nov. 2001. White paper, revision 2.6.

[9] VÄHÄAHO, M., HAATAJA, J.-P., METSO, J., SUORANTA, T., SILFVER, E., AND KUTVONEN, L. Pilarcos prototype II. Tech. rep., Department of Computer Science, University of Helsinki, Jan. 2003. C-2003-NN.