

# Open Source and Product Lines Architectures

Jesús Bermejo (1), Frank Golatowski (2), Francesco Furfari (3), Elmar Zeeb (2)

(1) Telvent, Spain

[jesus.bermejo@telvent.abengoa.com](mailto:jesus.bermejo@telvent.abengoa.com)

(2) University of Rostock, Germany

[frank.golatowski@uni-rostock.de](mailto:frank.golatowski@uni-rostock.de)

[elmar.zeeb@uni-rostock.de](mailto:elmar.zeeb@uni-rostock.de)

(3) ISTI-CNR, Italy

[francesco.furfari@isti.cnr.it](mailto:francesco.furfari@isti.cnr.it)

## Abstract

*The high level of customizability together with other business competitive advantages like reduced cost of acquisition and decreased vendor lock-in are pushing the open source adoption process. In parallel, the availability of the source code encourages the reuse. This process is evolving to an optimization of architectures that is meeting product line engineering. The product line maturity framework developed during ITEA-Families project identifies four key dimensions (business, architecture, process and organisation) to assess the maturity stage in product line development.*

*This paper analyses the impact of open source within the architecture dimension focussing on the middleware and variability support. It identifies major differences from the conventional approach in product line development introducing Osiris R&D platform as an effort for defining an infrastructure supporting product lines for multiple domains. It gathers preliminary results from open source related research from Cosiris [1] (ITEA-Cosi [2] & ITEA-Osiris [3] projects syndication) ITEA-Sirena [4] and Domoware [5] projects addressing foundations for ambient intelligence applications.*

## 1. Introduction

A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [6]. The competitive advantage of software product lines is currently worldwide recognised due to increased productivity, flexibility and customisation.

Frequently, open source development embeds an evolving architecture development process. The reuse of architecture assets differentiates product lines approach from other reuse strategies. In [7] the adoption of product line practises in several open source [8] initiatives is described by means of some examples. In this paper the product line maturity stages of ITEA-Families project [9] is taken as a reference to illustrate how open source is contributing to solve product line shortcomings in the development of major middleware platforms and having an impact on product line architectures.

The conclusions have been used in ITEA-Osiris (Open Source Infrastructure for Run-time Integration of Services) R&D platform, an effort for defining an infrastructure to support multiple product lines as foundation for ambient intelligence applications.

## 2. Influence of open source in the middleware layer of software product line architectures

The ITEA-Families project [7] identifies four dimensions (business, architecture, process and organisation) to assess the maturity stage in product line development. Table 1 defines these maturity stages from the product line architecture perspective.

The *lowest maturity level* is a situation of independent development of products. This stage is characterised by the absence of domain engineering in the development organisation. Only application engineering practises can be identified at this stage. Products are developed independently although ad-hoc reuse could exist.

| Maturity level in product line development              |   |
|---|---|
| Level 1: Independent Product Development                | No domain engineering (only application engineering). Products are developed independently although ad-hoc reuse could exist.   |
| Level 2: Standardised Domain Independent Infrastructure | Common software infrastructure (such as middleware or commercial off-the-shelf) is defined. Nevertheless, there is no formal reuse of domain specific assets.   |
| Level 3: Software Platform                              | Domain commonality is captured and implemented in a software platform. This platform is used for the different products. The platform could be configured. However, there is no variability support for product derivation. |
| Level 4: Derivable Variant Products                     | Domain commonality and variability is captured and a system family architecture is specified. Domain assets include support for deriving products.  |
| Level 5: Automated Product Derivation                   | Only Domain Engineering (no Application Engineering). Products can be derived automatically from the domain without product specific development.   |

**Table 1: Maturity level in product line development**

Figure 1 provides information about the use of programming languages in open source projects hosted by SourceForge. In this repository most of the projects run independently. Frequently, there are links among projects (in the repository or outside). Nevertheless, there is no systematic reuse based on the architecture. Developers in this forum are interested in reaching the users/developers with vertical applications reducing development cost and time to market.

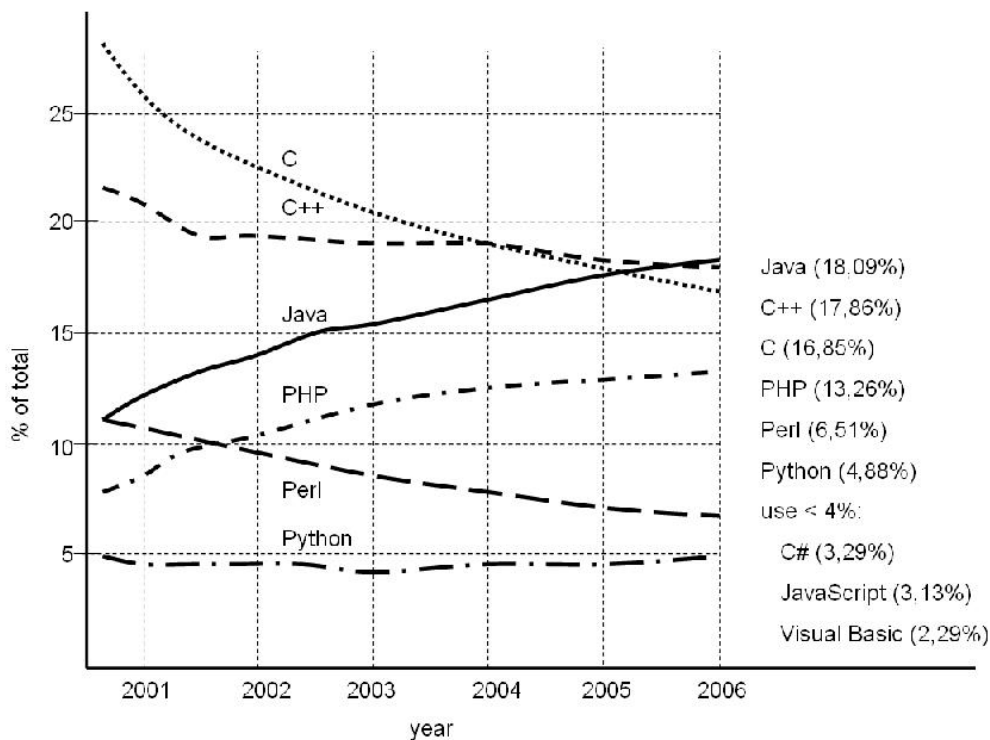


Fig 1. Programming languages use in SourceForge projects [8]

Fig 1 shows that the number of projects developed in Java is reaching those using C and C++ in the year 2006 [10]. PHP is following a similar trend with an almost parallel curve.

A *higher maturity level* in product line development is the adoption of a standardised domain independent infrastructure. In this level a common software infrastructure (such as middleware or COTS) is defined for the product line but there is no formal reuse of domain specific architecture assets.

It is not strange the relationship between the most used languages in open software projects and the success of middleware communities developing middleware in these languages (e.g. Apache, Eclipse, Java.net, JBoss, Objectweb, Codehaus, Spring, Opensymphony, for Java and PHP.net/pear, Zend framework related to PHP). This can be justified due to the fact that the middleware addresses non-functional requirements which to some extent are transversal commonalities across systems from multiple domains.

In a conventional scenario of proprietary development the domain independent functionality is frequently covered with commercial middleware. Reference platforms in this segment market addressed by specialised organisations have importantly increased in size and complexity during the past years. The difficulty in addressing optimised product lines in this layer (embedding the business dimension) has pushed the market towards very large middleware software platforms (such as J2EE application servers) covering most of the needs. However, an important part of the functionality provided is not required for specific applications in which this middleware is used. This approach simplifies the product line business for a middleware provider that can deliver a single platform (or with limited variability) covering requirements for multiple customers.

The open source implementations of the high-end middleware platforms have frequently reused existing open source projects. On the other hand, once they have become popular, the possibility of re-using parts for other projects is leading to re-factoring processes. The availability of the source code seems leading a re-use process which is evolving to its optimization. This is reinforced through a distributed development organisation that encourages the adoption of architectures facilitating a parallel and independent development.

The increasing intelligence in embedded systems is starting to play a key role in this process once Java programming is becoming popular in embedded devices. The need of high customizable platforms is pushing the adoption of light-weight frameworks. The flexibility provided by these frameworks seems to be suitable to support the use of parts covering with this the requirements of enterprise middleware platforms too.

The difficulties in handling optimised product lines for large middleware platforms with the conventional software development approach seem to be overcome with the open source approach. This has important architecture implications for the domain layer as the boundaries between middleware and domain will become much more diffuse. ITEA-project COSI [1] addresses these integration issues.

A *higher maturity level* in product line development entails capturing the commonality in the domain to be used for different products. In this maturity level variability and supporting mechanisms are key assets.

The possibility of a high customisable middleware is a new approach brought by the open source. Light weight frameworks provide mechanisms for supporting only the required middleware components. The same mechanism can be used for the domain layers. This facilitates deriving products embedding only the required combination of middleware and domain components. Nevertheless, these light frameworks are oriented to stand alone computing nodes when many software intensive systems are moving to a more distributed architectures.

A new context of computing devices linked together through a variety of networks (xDSL, Cable, GPRS, UMTS, Satellite, WiFi, Bluetooth, ...) is moving market segments needs beyond the boundaries of single computing nodes. The functionality derived from the cooperation of systems is everyday more important. In fact, frequently, overlapping systems can be defined in terms of different system boundaries. The increasing need of interoperability among systems is pushing Service Oriented Architectures (SOA) [11]. Thinking in terms of something to be used (i.e. service) rather than in terms of something that "is" (i.e. component) has important implications in the product lines architectures in order to select the adequate mechanism to support the variability. SOA entails a higher level of abstraction that does not worry about how a service is implemented. The support of service directories can provide additional level of independence from the location of the execution even allowing changes during run time. In networked environments SOA programming principles increase the flexibility of the architectures for building product lines.

### **3. The OSIRIS R&D platform**

Previous rationale has driven basic concepts for the OSIRIS platform which is targeting an infrastructure for multiple product lines as foundation for ambient intelligence applications. The OSIRIS platform is a distributed, service-oriented platform for product lines with the following elements:

- A network of OSIRIS nodes. OSIRIS nodes can play different roles as service consumer or service provider.
- Software asset repositories. Networked entities that enable deploying software packages to OSIRIS nodes. Repositories are also docking stations where software providers upload software packages representing services implementations.

- Service directories. A service directory provides services metadata and search capabilities. Metadata can include references to software asset repositories in case software packages are required before accessing a service.

OSIRIS nodes are the main physical elements of a physical platform. An OSIRIS node is basically a device powered with a networked runtime environment. It may play more than one role and thus, depending on the particular scenario, may have different needs. In order to address the diversity of potential uses a basic profile has been defined (a minimum set of capabilities) that afterwards can be tailored to each scenario by means of profile extensions. Profile extensions provide specific functionalities in addition to the basic profile. A profile extension is a logical unit that adds some functionality or capability to an OSIRIS node. Profile extensions are packaged and deployed as groups of OSGi bundles.

An OSIRIS node comprises an OSGi framework and a basic profile installed. Thus, nodes are service-oriented platforms with support for remote deployment and life-cycle management of components. This also implies that the deployment unit for OSIRIS nodes is an OSGi bundle or a set of related bundles. The variability is defined in terms of two levels of granularity; one associated to the deployment unit and the second to its configuration.

The node is able to expose and access remote services. The remote services communication component provides different remote protocols to interoperate with the OSIRIS network. OSIRIS nodes can collaborate among them and provide access to non OSIRIS nodes such as sensors or actuators.

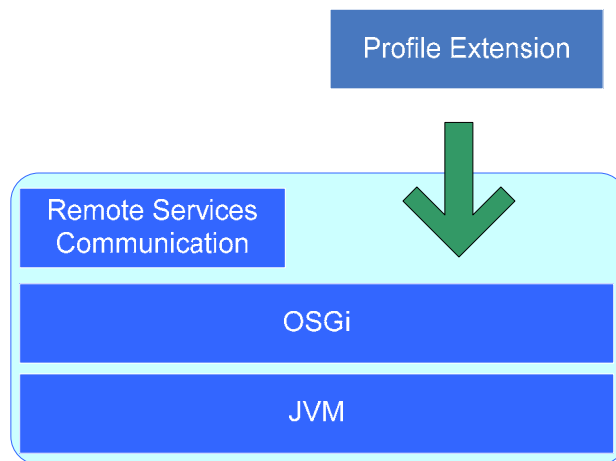


Fig 3: OSIRIS Basic Profile

#### 4. Product lines involving several nodes. WS4D and Domoware

This section describes state of the art of SOA technologies usable for networked OSIRIS nodes. The services for collaboration among nodes support the variability of the system associated to the existence of physical nodes (e.g. optional nodes). System boundaries (i.e. limits of the nodes that are part of a system) can be defined by a distributed authorization and authentication framework.

Different DPWS solutions offered by the WS4D- Initiative [12,13] and one to cover Universal Plug and Play (UPnP) offered by the DOMOWARE-project are available to support the variability required for the foundation of ambient intelligent applications involving nodes with different computing capabilities. (one for Apache AXIS2, one for gSOAP (C,C++) and one for J2ME). The concept will be validated against scenarios in multiple domains.

#### **4.1. Combining UPnP and OSGi for an easy application development.**

The Domoware Project [5] was the first Open Source initiative to provide a public implementation of UPnP specification of OSGi Release 3. Many research projects addressing autonomic computing, pervasive/ubiquitous computing and ambient intelligence have tested and positively experienced the use of OSGi/UPnP specification [17,18]. The dynamic nature of the UPnP networks, in which networked devices can be plugged and unplugged at any moment, has been proved to be consistent with the capabilities of the OSGi platform of handling run-time variability. UPnP Devices, sensors and actuator are easily bridged in the OSGi platform where refining drivers can be automatically installed from software asset repositories [19,20].

Alternative uses of the UPnP/OSGi specification has been also proposed and experimented in these years. For example, the basic profile of OSIRIS nodes, or generic OSGi containers, could be augmented with a custom UPnP device in order to implement the dynamic discovery of OSGi platforms. Recently a charter proposal to the UPnP Forum has been submitted by FranceTelecom and Samsung in order to standardize an UPnP device profile for managing execution platforms like OSGi.

#### **4.2 Using UPnP and JMX to improve manageability of OSGi**

Another notable approach for the collaboration among nodes is the combination of JMX, OSGi and UPnP technology. Remote management of set of OSGi gateways by using JMX probes, according to one of the business model of OSGi Alliance, is addressed in Apache subproject Felix. They are installed by bundles and allow a management centre to monitor and control services running on managed gateways. By writing a probe for the UPnP specification, UPnP devices running on the local networks of managed gateways can be connected and interoperated with other UPnP devices installed on the management centre network. Also in this context the OSGi deployment model is a paramount because the variability of the controlled environments requires systems to be configured and adapted dynamically [21].

The adoption of the device/service model devised by UPnP specification as a general model for remote access to OSGi services has not been solved. Generic OSGi services should implement UPnP service model. Even if code tagging and injection could simplify the process, UPnP offers only limited support for complex data types

#### **4.3. Using Web Service based approach**

An alternative solution to UPnP is the Devices Profile for Web Services (DPWS) that was specified to enable secure Web service capabilities on resource-constraint devices. DPWS is a very recent specification based on up-to-date Web services specifications and is integrated into Microsoft's latest operation system Vista. The European R&D ITEA-SIRENA project (Service Infrastructure for Real-time Embedded Networked Applications) has successfully used this technology as basis of its infrastructure and has shown its feasibility in the areas of industrial automation, home entertainment, automotive systems and telecommunication systems. When the SIRENA project ended in the beginning of 2006 some partners continued their work on the different DPWS toolkits, to obtain their interoperability and to bring them to open source. The combined efforts can be found at the WS4D [12] initiative and the SOA4D [14] forge. At the moment there are several solutions for developing devices being conform to DPWS. Since we assume that future device ecosystems will be heterogeneous systems, the toolkits are targeted for different languages and different platforms of different scale [13].

In case of multiple computing nodes there are several possibilities to integrate DPWS into the SPL. The most obvious solution is to integrate DPWS into OSGi. The integration can be implemented in the same way as with UPnP, as DPWS is comparable to UPnP. This would enable access to services on DPWS-based devices from within OSGi. There is already ongoing work to integrate DPWS into OSGi Release 4 [15]. DPWS could also be used to access OSGi services, as it doesn't have as much restrictions on the messaging layer concerning the

message structure complexity as UPnP [16]. But it is still arguable if technologies like SOAP and XML etc. used in DPWS are the best choice for message exchange with OSGi services.

Another approach is to use DPWS directly as platform for device centric SPLs. This approach does not exclude OSGi but utilizes mechanisms specific to DPWS and Web services. The Devices Profile for Web Services offers type systems for devices and services hosted on devices. These type systems enable definition of specific functionality on device and service level that can be implemented as components and be reused on subsequently developed devices. A similar type system is available in UPnP. Workflow technologies offer new opportunities to distributed SPLs. With technologies like WS-BPEL new products can be developed above the device and service level.

## 5. Conclusions

The difficulties in handling optimised product lines for large middleware platforms with the conventional software development approach has led to monolithic and large middleware platforms. The problems seem to be overcome with open source approaches. The possibility of reusing the code combined with the existing light weight frameworks originally targeting embedded systems are providing the foundation for a new approach to optimised middleware.

The tight coupling of middleware and domain assets allowed by the new architectures can make advisable the use of the same mechanisms to support the variability both in the middleware and in the domain.

A new context provided by the communication infrastructure is moving market segment needs beyond the boundaries of single computing nodes. SOA principles bring powerful principles for supporting the building of product lines involving several nodes.

Based on the new approach to middleware the OSIRIS R&D platform targets an infrastructure for multiple product lines as foundation for ambient intelligence applications. The solutions for variability should address both single and multiple nodes product lines. The concepts are being developed in the OSIRIS platform which will be demonstrated for several domains. Open source implementations of several relevant technologies such as OSGi framework, UPnP, DPWS and solutions for providing Web Services support to nodes with different computing capabilities are available.

As a global conclusion, the open source strengths for shaping product lines practices seems contributing to more mature approaches than those in the conventional development scenarios with independent middleware and domain software assets providers. This seems to lead a vision of a convergence between these two fields as a direct consequence of the common driving principle; optimising software development through optimum reuse.

## References

- [1] COSI: Co-development using inner & Open source in Software Intensive products, <http://www.itea-cosi.org>, 2007
- [2] OSIRIS: Open Source Infrastructure for Run-time Integration of Services, <http://www.itea-osiris.org>, 2007
- [3] COSIRIS: COSI & OSIRIS, <http://www.cosiris.org>, 2007
- [4] SIRENA: Service Infrastructure for Real-time Embedded Networked Applications. <http://www.sirena-itea.org>, 2006
- [5] DOMOWARE, Institute of Information Science and Technologies (ISTI), <http://domoware.isti.cnr.it>, 2004

- [6] Software Engineering Institute, Carnegie Mellon University,  
<http://www.sei.cmu.edu/productlines/>
- [7] Jesús Bermejo, Naci Dai: Open Source Strengths for Defining Software Product Line Practices, First International Workshop on Open Source Software and Product Lines, Baltimore, Maryland USA, 2006
- [8] OSI: Open Source Initiative, <http://www.opensource.org>
- [9] FAMILIES, <http://www.esi.es/Families/>, 2005
- [10] François Labelle, Programming Language Usage Graph,  
<http://www.cs.berkeley.edu/~flab/languages.html>
- [11] M. Treiber, S. Dustdar, An overview of Service-Oriented/Aware Middleware, Whitepaper, Technical University of Vienna, 2006  
<http://www.infosys.tuwien.ac.at/staff/treiber/Overview.pdf>
- [12] WS4D Initiative, <http://www.ws4d.org>, 2006
- [13] E. Zeeb, A. Bobek, H. Bohn, S. Prüter, A. Pohl, H. Krumm, I. Lück, F. Gólatowski, D. Timmermann, WS4D: SOA-Toolkits making embedded systems ready for Web Services Open Source Software and Productlines 2007 (OSSPL07), Limerick, Ireland, Juni 2007
- [14] SOA4D Forge, <http://www.soa4d.org>, 2007
- [15] A. Bottaro, A. Gérodolle, S. Marié, Combining OSGi™Technology and Web Services to realize the Plug-n-Play Dream in the Home Network, OSGi Alliance Community Event, Munich, June 2007
- [16] Jan S. Rellermeyer, Gustavo Alonso: Services Everywhere: OSGi in Distributed Environments. In: EclipseCon 2007, Santa Clara, CA, March 2007
- [17] D. Sacchetti, Y. Bromberg, N. Georgantas, V. Issarny, The Amigo Interoperable Middleware for the Networked Home Environment, 6th International Middleware Conference, Workshops Proceedings, Grenoble, France, 2005
- [18] Tao Gu, Hung Keng Pung, Da Qing Zhang, Toward an OSGi-Based Infrastructure for Context-Aware Applications, IEEE Pervasive Computing, vol3, no.4, oct. 2004, pp 66 - 74
- [19] D. Donsez, On-Demand Component Deployment in the UPnP Device Architecture Consumer Communications and Networking Conference, CCNC 2007 IEEE Jan. 2007 page(s):920-924 Las Vegas, NV, USA,
- [20] N. Goeminne, K. Cauwel, F. De Turck, B. Dhoedt, Deploying QoS sensitive services in OSGi enabled home networks based on UPnP. In: 2006 International Conference on Internet Computing - ICOMP. 2006. (26-29 June 2006; Las Vegas, NV, USA.)
- [21] S. Frenot, Managed OSGi framework, Apache Felix Project,  
<http://cwiki.apache.org/FELIX/mosgi-managed-osgi-framework.html>