A New Development Methodology and Reference Architecture for Virtual Environment Systems

Manuel Oliveira  
University College London  
Department of Computer Science  
Gower Street, WC1E 6BT London, UK 
m.oliveira@cs.ucl.ac.uk

João Pereira  
INESC  
Rua Alves Redol  
1000-029 Lisbon, Portugal 
jap@inesc.pt

Abstract. In an age where software permeates our lives, software engineering plays a crucial role in the life cycle of digital systems. Although object-oriented programming languages have begun to be adopted in the development of virtual environment (VE) systems, the traditional software process does not apply software engineering to address the complexity of the problem domain.

This paper presents the Virtual Environment System Layered Object Model (VESLOM) as a new development approach that addresses the complexity of the problem domain. The VESLOM approach consists of a design process combined with a reference architecture for building VE systems. The paper also presents the results of a formal evaluation of Java Adaptive Dynamic Environment (JADE) as the proposed Universal Platform, the core of the VESLOM reference architecture.

1. Introduction

The wide proliferation of systems supporting Virtual Environment (VE) clearly demonstrates the impossibility of developing a single solution that meets a broad range of possibly diverging requirements. In fact, many of these requirements present themselves as conflicting and contradictory to one another. This is a direct result of the complexity of the problem domain associated to VE systems. However, the traditional development trend remains unchanged and unable to address the many limitations that have been duly recognised to various degrees by the VE developer community.

A careful analysis of the different systems shows some signs of maturity in the development process. As illustrated in (A) of Figure 1, the primordial solutions were monolithic with unclear boundaries delineating parts of the systems that dealt with specific functionality. The current status of existing systems is portrayed by (B) of Figure 1 where it is possible to delineate subsystems with a particular role and functionality. Although the current systems have become more flexible, the result continues to be monolithic and scoped to a particular subset of the VE problem domain.

Figure 1 - Flexibility within VE systems. The outer boundary of the system is encompassed by a black line. (A) Unclear subsystems. (B) Clear subsystems

The most recent systems have adopted object-oriented development languages, but continue to be constrained by the pitfalls of the traditional development process. Therefore, it is common to have a new VE system each time there is a change in the initial system requirements. This is unlike the perspective of software engineering community that consider it a rarity for a large complex system to be developed from scratch [9].
This paper presents the Virtual Environment System Layered Object Model (VESLOM) as an alternative to the development process of VE systems. The paper is structured into six further sections, starting with an overview of the pitfalls common to the current software process in section two. The VESLOM Design Process and reference architecture are presented in sections three and four respectively. A proposal for the Universal Platform is presented in section five with an evaluation given in section six. Finally the paper provides some conclusions in section seven.

2. Overview of the Traditional Development Process

The field of software engineering emerged to address the concerns and difficulties involved in developing large complex systems to be deployed in the real world. A VE system fits this category, but little or no systematic methodology is employed in the software process. In fact, the design phase in the common software process is very much inexistent or extremely short with a bare sketch of the overall architecture before proceeding to the actual implementation. Therefore the software process can be characterized as ad-hoc and chaotic.

When assessing the outcome of the current software process concerning VE systems, one is confronted with a wide proliferation of different solutions, each aiming to solve a specific sub-domain model. Since the system is disregarded as a whole based on a combination of equally important elements, the result is an unbalanced system that excels at specific functionality whilst performing poorly in all the remainder operational areas. This is clearly illustrated in the case study of [23], where the Distributed Interactive Virtual Environment (DIVE) [10] system could not fulfil the requirements of collaborative haptic experiment across a transatlantic link due to fundamental system design choices, namely the fact that the graphics rendering cycle determined the systems processing frequency ($\approx 60$Hz). This constraint made it unfeasible to meet the requirement to support the 1 KHz of the haptic rendering cycle. Since it was not possible to make the necessary modification, the result was a new system developed from scratch, particularly tailored to fulfil the requirements of the experiment to carry out and with limited reusability.

A common architectural characteristic of existing VE systems is their monolithic nature. This prevalent inherent characteristic reduces the flexibility of the system and imposes serious constraints on extension to accommodate shifts in the associated sub-domain model. Some characteristics, namely the system’s lifecycle, inflexibility, scalability, network and the influential human factors to the software process, provide strong evidence that the development trend either remains unaware or neglects to accommodate any mechanisms to address the issues linked to the software evolution and aging phenomenon.

Although the approach of the current development trend is very much ad-hoc, there are a few exceptions where software engineering principles have been adopted since the inception of the system and shaped the design to maximize flexibility. The Maverik system [11] is such as system, where the concern is to provide a framework that is independent of the rendering data model, thus different rendering techniques can be adopted and adapted to the particular needs of the application. Although the main aim of a flexible framework for the rendering pipeline that is independent of the supporting data structure and particular algorithms, the system is unbalanced. In Maverik, the design choice was to make it standalone without networking capabilities. It is necessary to rely on DEVA [25] for the distribution infrastructure of a VE. However, even though both projects have been developed in parallel within the same research development team, their integration suffers from serious interoperability problems.

The crux of the problem resides with the top-down approach of the traditional development process. This leads to the selection of core concepts with high semantic connotation that shapes the architecture, influencing every aspect of the system. An important lesson derived from a decade of building reusable communication and application frameworks [26] is the realization that generalization leads to successful reusability when emerging from a bottom up approach analysis of various system instantiations of frameworks.

3. VESLOM Design Process

Irrespective of the development methodology adopted in the software process of a VE system, the VESLOM Design Process provides guidelines to improve the reusability and interoperability of the
final solution. The VESLOM Design Process is based on the OIA/D method [18] as evidenced in the schematic diagram of Figure 2.

As with OIA/D, the process subdivides itself into two distinctive phases:

- **Analysis.** This phase focuses on building the model of what lies within the atypical blackbox of a component. The process begins by using the domain knowledge to establish and refine the service interface of the component, taking into account the various usage profiles. With the service interface defined, the analysis proceeds to model the invariant implementation structure of the component isolating the potential implementation dilemmas, thus identifying the various strategies involved.

- **Design.** This phase transforms the model resulting from the analysis phase and builds the corresponding design, taking into account the knowledge from the problem domain, garnered from experience and the design artefacts (i.e., patterns). It is during the design that the Meta Object Protocol (MOP) is built to allow manipulating the internal implementation of the component.

As illustrated in the Figure 2, a component can be designed in such a way that only the service interface is established, without any internal implementation structure, thereby requiring the selection of a strategy before any usage can take place. The selection process does not imply that the strategies are pre-existent within the component, but may be provided via the MOP interface.

The VESLOM Design Process integrates core principles from the agile software development community [3] that promotes a light-weight software process based on iterative short cycles where software reengineering is deeply ingrained throughout. The lean development philosophy is achieved by reducing the requirement of supporting documentation and relying on best practices to ultimately increase the productivity. However, an analysis [27] of the foundational principles,
along with the necessary assumptions made about the software process, reveals that there continues to be no such thing as “silver bullets” [5]. In addition, the existing agile methods still require some maturity [1] in order to address their limitations, namely the disregard to generalization to address the issues of reusability and lack of practical guidance to applicability.

Despite the apparent anathema to design, the VESLOM Design Process incorporates some of the practices from agile programming, namely refactoring [19], which consists on restructuring the internal implementation of a component without having an impact on the external service interface. In VESLOM, the restructuring is not only applied during the implementation phase, but also to the model and design phases.

4. VESLOM Reference Architecture

The result of applying the VESLOM Design Process to address the complexity of the problem domain of VE yielded the reference architecture of Figure 3. Similar to the Virtual Reality Transport Protocol (VRTP) [6] proposal, there is the concept of a Universal Platform at the core of the reference architecture. However, rather than delving into defining specific components and enforcing a particular implementation, the approach taken was to adopt a layered object model with total separation between the architecture and the implementation.

The paradigm of using layers addresses the complexity associated to the problem domain, but breaking the coupling between layers to reduce the monolithic nature of a VE system depends upon constraining the dependencies between layers. As expressed in [2], any dependency between a layer and the layers above defeats the advantages facilitated by system architectures. Therefore, to avoid coupling, if a layer needs to communicate with an outer layer, then an appropriate mechanism is used, such as an event model.

![Figure 3 - VESLOM Reference Architecture](image)

The Universal Platform provides the common minimal infrastructure to all VE systems, but the inclusion of network capabilities is a design pitfall to be avoided since it is not possible to determine the ideal network subsystem and it should be deemed optional for the cases where a system works standalone. Therefore, VESLOM provides an additional abstraction layer that is responsible for networking support allowing the possibility of leveraging the application knowledge to shape the data flows (TreaileWell [22]). The middleware layer encompasses all the components with well-defined functionality to be reused in different systems via a common interface. Finally the application layer relates to all the non-generic components or software artefacts that are tightly coupled to the particular instantiation of the VE system.

Contrary to other layered architectures for VE systems, VESLOM delegates to the stakeholders the responsibility of choosing what components to employ as part of a particular solution. Therefore, if all the layers of VESLOM are adopted, along with all the proposed components, the application layer practically presents a working system. This approach would only require some customization and modeling of the VE, but it is not expected for the stakeholders to manipulate any of the components that are part of the underlying system. The other extreme of the development approach, would consist of only adopting the Universal Platform, thereby abolishing any of the constraints associated to the existing components. However, while this option maximizes the
flexibility within the software process, there is significant resource expenditure to attain a solution that can be used.

5. Universal Platform

The challenge of developing the Universal Platform is to determine from the problem domain what is common across any VE system and condense the result of the analysis into an extensible architecture that has the potential to support all manner of functionality. A similar challenge was addressed in the field of operating systems, where the kernel comprises of the essential part that is common to any software. The primal design, as with VE systems, was based on monolithic architectures that raised serious problems such as flexibility, extensibility, reliability and maintenance. In response to the drawbacks, new architectures emerged, namely the concept of micro-kernel that consisted on a minimal kernel where the services and policies are delegated to the user space of an operating system. The first generation of micro-kernels (ie: Choices[7]) produced disappointing results since their utility was compromised by the poor performance and failure to adhere to the core design principles of small, simple and flexible kernels. A careful analysis [16] demonstrates that the fallacy related to poor performance is due to poor selection of implementation strategies. This realisation led to a second generation [17] of micro-kernels (ie: Exokernel [8]) where major restructuring was done to increase the performance whilst addressing the shortcomings in the design. To address the concerns of flexibility, some solutions, such as Apertos [29], have adopted open implementation (OI) as a foundational design paradigm, but the associated performance penalty to the approach is a significant deterrent for wider adoption.

The Universal Platform provides a micro-kernel that is the foundation of any and all VE systems, but the responsibility of managing hardware resources is delegated to an existing operating system without any particular coupling constraint. This implies a mechanism similar to the Spindle mechanism of the SPIN micro-kernel [4], which supports run-time operation, with the former being responsible for loading and unloading of modules into a system whilst the latter is a mechanism to compile and integrate code into the kernel.

The application of the VESLOM Design Process, to determine the Universal Platform, resulted in the Java Adaptive Dynamic Environment (JADE) [24]. The genesis of JADE came as a direct result of the analysis [20] done to documentation available concerning VRTP, namely the proposed solution for their Universal Platform - Bamboo [28]. Although the concepts put forth held promise concerning the way VE systems are built, the primary focus degenerated into the development of a unique solution to be adopted by all. This would be unfeasible considering the amount of implementation dilemmas that would raise issues of contention that would deter any wide adoption of a single implementation solution. Therefore, in JADE, there is a total separation of between the reference architecture and the reference implementation (component framework).

Table 1 – Comparison between Jamboo and JADE in terms of (I)nterfaces and (C)lasses according to the Reference Architecture (RA and the Component Framework (CF). The packages in red denote a direct mapping to the building blocks of the RA.

<table>
<thead>
<tr>
<th></th>
<th>RA I</th>
<th>RA C</th>
<th>CF I</th>
<th>CF C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jamboo</td>
<td>3</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JADE</td>
<td></td>
<td></td>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td>Jade</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>jade.compiler</td>
<td>1</td>
<td>6</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>jade.event</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>jade.generic</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jade.kernel</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>jade.locator</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>jade.namespace</td>
<td>5</td>
<td>3</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>jade.security</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jade.util</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>24</td>
<td>71</td>
<td></td>
</tr>
</tbody>
</table>

To determine the main building blocks part of the JADE’s system architecture it was necessary to eliminate all abstractions with the slightest semantic connotation to a process or method not common to all VE systems. Jamboo was the first incarnation [21] of the JADE architecture core.
sharing strong similarities with the Bamboo [28] initiative, but focused on addressing the identified fallacies. As reflected in Table 1, the initial architecture consisted only of Module management with a total of 17 classes and three interfaces. However, there was no separation between the architecture and the reference implementation. As the design was tightly coupled with particular implementation strategies, the result was a solution that was more flexible than existing solutions, but contention would compromise adoption due to the mapping dilemmas.

The VESLOM Design Process contributed to the decoupling of the Reference Architecture from the component framework of the reference implementation as evidenced in Table 1. The resulting JADE architecture is illustrated in Figure 4, where the functionality is decomposed into five main building blocks that are necessary to any VE system:

- **Namespace.** A framework for the hierarchical management of resources, including retrieval operations using filters and rules. Two reference implementations available, one based on memory registry and another based on a relational database. The micro-kernel has a namespace for management of code, but namespaces are used throughout the middleware components, namely the Meta Unified Datamodel (MUD) that manages geometry and behaviour content of a virtual environment;

- **Resource Locator.** The resource locator is an extensible component that has a set of retrievers and handlers. The former are responsible for retrieving the raw data from any location using a specific method (ie: http, ftp, jini™, file system, etc). The latter are responsible for handling the raw data, thus code is dispatched to the runtime linker of the micro-kernel, images are stored in default file location and trigger an event, configuration files trigger an event for the Configuration building block, etc. In the case of code, the dynamic linker is responsible for resolving any dependencies and these are solved based on the default and current codebase;

- **Configuration.** The configuration is a simple interface that notifies when configuration is complete. The micro-kernel adopts the configuration and combines it with a compiler component framework which can be extensible at runtime as needed. Each module loaded at runtime can indicate the configuration mechanism used, namely build the compiler and indicate the configuration file;

- **Event Model.** The need of communicating within a virtual environment system is supported by two types of event model. A distributed model, which delegates to each particular source the responsibility of managing the listeners. A centralized model, which consists of an event manager where listeners register their interest in particular events and/or sources, using if necessary filters and priorities. The events can be either asynchronous or synchronous;

- **Security Manager.** The security manager uses a fine grained policy mechanism that oversees and enforces the access rights to each resource and operation. There is a default security policy used in case none is provided.

The current architecture has achieved sustained stability reflecting its maturity that resulted from the evolution process affecting JADE through several projects, totalling eleven packages, 72 classes and 42 interfaces, distributed according to Table 1 (the packages in red have direct mapping to the building blocks). The key concepts survived all the iterations, with the architecture becoming more refined. However, from the implementation perspective, the component framework did suffer
drastic modifications on occasions, such as the introduction of the concept concerning namespaces that had a rippling effect throughout the reference implementation codebase.

The reference architecture is independent of any implementation strategies and the choice of implementation language. Although the component framework adopted the Java programming language, there are no constraints that prevent alternative options. The main reasons for adopting Java as the development language for the codebase was to leverage the increased productivity, the strong object-oriented features and the inherent functional capabilities of the Java Virtual Machine (JVM), such as the dynamic linking at runtime which is necessary to satisfy some of the UP scenarios.

6. VESLOM Evaluation Method

A foundation of the entire VESLOM approach, from the design to the evaluation is the use of scenarios to identify the quality attributes, both qualitative and quantitative, that must be embodied in the system architecture.

The VESLOM Evaluation Method (VEM) is an extension from the Software Architecture Analysis Method (SAAM) [13], combining together elements from the Architecture Tradeoff Analysis Method (ATAM) [14]. The fundamental difference between other methods and VEM is evident in purpose of employing the method. In all methods, the participants can be categorized into two main groups:

- **Evaluators.** These participants are the ones that will drive the evaluation based on the adopted method and by interacting with the stakeholders;

- **Stakeholders.** These participants have a vested interest in the architecture and the resulting particular instantiation into a system. There will be different classes of stakeholders, but usually the classes constrain themselves to the developing and managerial teams.

Although VEM has the same two groups of participants, the difference emerges regarding the stakeholders population. In the other methods, the stakeholders are members of a single company or an organization, each with particular visions for the final solution. Consequently, the stakeholders are only interested in a single instantiation of the architecture to meet their needs in the immediate and near future. In VEM, the aim is to design an architecture that addresses the problems of the virtual environment community, thus the stakeholders belong to general population, companies and organizations. This approach contributes to the adoption of the Universal Platform by reaching an architecture that supports the widest range of critical case scenarios from the community, where each has differing specific visions for a virtual environment system.

6.1. VEM Description

The VEM aims to evaluate the Universal Platform, but also to be a mechanism to impart the knowledge of the supporting architecture to the stakeholders of the community and align their expectations. The process consists of 10 well-defined steps aggregated into 4 phases as illustrated in Table 2.

The questionnaire from step 2 consisted of a questionnaire with 24 questions. The aim was to determine some of the experience of the stakeholder and their inclination towards software quality and their awareness to the problems that plague the traditional development process.

The questionnaire from step 9 consisted of 16 questions. The aim was to collate qualitatively their assessment of the material presented, VESLOM and JADE. In particular, four questions were targeted at qualify their predisposition to adopting the JADE reference architecture and taxonomy in initiating and future projects.

In both questionnaires, the responses were based on a Likert scale, between 1 and 7. The lower-end of the scale normally corresponds to when the respondent has a negative response to the question, whilst the upper-end of the scale indicates that the respondent has a positive response to the question. The questions were carefully structured on a qualitative approach since it would be...
difficult to homogenise a quantitative assessment across all stakeholders. With the data collected, the responses were aggregated together into one of the three classes:

- **Negative.** This class corresponds to when the respondent has a negative answer concerning the question. This implies that answers 1, 2 and 3 are aggregated together.

- **Neutral.** Normally, people adopt the middle of the scale when indecisive or if they do not have a strong opinion on the question. This is corresponds to 4.

- **Positive.** This class corresponds to when the respondent has a positive answer concerning the question. So answers 5, 6 and 7 are aggregated together.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Designation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>1</td>
<td>VEM Process</td>
<td>This involves giving an overview of VEM, describing what it entails and the purpose of the process. This step will also involve the presentation of relevant documentation concerning the Universal Platform to convey to the stakeholder a clear understanding of the architecture.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Stakeholders Profile</td>
<td>This involves the stakeholder filling a small questionnaire that helps identify their profile.</td>
</tr>
<tr>
<td>Analysis</td>
<td>3</td>
<td>Vision</td>
<td>This step presents the vision that drives the VESLOM Reference Architecture (RA). This involves conveying the complexity of the problem domain associated to virtual environments, which leads to the current proliferation of different systems. The stakeholder will be presented with some of the most prominent pitfalls inherent in current development process. Derived from the problems and pitfalls, a set of objectives will be delineated that define the VESLOM RA, with particular focus on the JADE as the Universal Platform architecture.</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Scenarios</td>
<td>This step consists of the description of the main scenarios to be supported by the architecture. These scenarios are described in the necessary detail to capture the quality attribute whilst eliminating the ambiguity.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Architecture</td>
<td>This step entails the presentation, in sufficient detail, of the underlying architecture of JADE as the Universal Platform. This will involve the description of the main building blocks and their operation. The result will convey a clear understanding of the architecture to the stakeholder taking into consideration the vision and main scenarios identified.</td>
</tr>
<tr>
<td>Brainstorm</td>
<td>6</td>
<td>Scenario Revision</td>
<td>This step involves to revision of the scenarios identified in step 4 taking into account the context of the stakeholders participating in the VEM. This allows them to revise the existing scenarios and add new ones. After the identification of a set of requirements, the VEM process proceeds with the prioritization of the scenarios, selecting the most important ones.</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Architecture Discussion</td>
<td>This step will consist of discussion of the architecture to validate if the final set of scenarios of step 6 can be supported by JADE. The result may identify some architectural changes.</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Evaluation</td>
<td>This step consists of an overall evaluation by the stakeholder of the architecture in fulfilling the initial scenarios and the flexibility to support the revised scenarios.</td>
</tr>
<tr>
<td>Synthesis</td>
<td>9</td>
<td>Stakeholder Assessment</td>
<td>With this step, the stakeholder is asked to fill another questionnaire that will capture qualitatively their evaluation of the JADE.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Final Assessment</td>
<td>This step wraps up the process by collating all the information generated from the VEM process and synthesizing the results. The resulting assessment will determine the success of the Universal Platform in fulfilling the proposed vision.</td>
</tr>
</tbody>
</table>

6.2. VEM Evaluation

The VEM process was carried out with six individuals from different classes of stakeholders and belonging to different organizations. Each assessment consisted of an individual working session with each of the stakeholders, whose profile ranged from virtual environment system developers to a company manager. The shortest session lasted 105 minutes, whilst the longest went beyond 310 minutes, but the median was 135 minutes.

Independently of the specific profiles of each individual, the questionnaire (step 2) denoted common characteristics amongst the stakeholders. They all evidenced significant experience in the
development of virtual environment systems, with more than eight years and leading projects involving system development. Their background was either from computer science or engineering. With the exception of one stakeholder, all agreed that software engineering techniques are beneficial, but in reality little is actually put in practise with reusability being a much desired elusive goal to achieve. The only agreed upon reusability technique would be code scavenging.

The maturity of JADE was evidenced by the fact that no architectural changes were identified and any new scenario was either foreseen or easily accommodated without any modifications. All stakeholders approached the problem of stressing the architecture by considering how their current and future work could be supported by JADE. Considering the varied background of the projects, from crowd simulation, online games, avatars, collaborative virtual environments, research in presence and co-presence, etc, JADE demonstrated to fulfil all the necessary requirements of different sub-domains of the problem domain associated to VE.

The responses to second questionnaire (step 9) was overwhelming positive (most responses were 7 with a few giving 6) in the assessment of the materials, VESLOM reference architecture and JADE as the proposed Universal Platform. This was evidenced in the keen interest to have further information and more details concerning the taxonomy and architectural guidelines for potential use in future projects involving the development of a virtual environment system.

7. Conclusions

There is a proliferation of VE systems which share common functionality but the solutions have little commonality to one another. The reigning monolithic nature of the various systems makes it difficult to tailor a particular solution to a subset of the problem domain. The crux of the problem is the complexity of the problem domain and the ingrained approach of building VE systems based on a top-down perspective. Taking into account that development resources (time, manpower and budget) are limited, the result is unbalanced systems. These solutions reveal some areas within the system that excel at the prime design aims whilst demonstrating weaknesses, ranging from minor to grave, at other areas not considered as important.

Although much of the software process for VE systems continues to be prominently Ad Hoc, sound software engineering principles begin to permeate the software process of some solutions. However, the end systems are still plagued with monolithic architecture that cannot accommodate radical changes to the original base functionality, albeit the improved internal flexibility. The problem worsens when considering that all solutions lack any form of interoperability with the exception of geometry exchange due to standards such as VRML and X3D.

A simplified version of VEM was devised to be part of the VESLOM Design Methodology. The method resulted in many discussions with members of the wider VE community. The rich interaction led to the refinement of the scenarios representing the functionality to be supported and consequently the maturing of the reference architecture. However, the process has increased the awareness of JADE, which has led adoption of terminology and concepts in the design of new systems, thus easing the issues pertaining interoperability, such as in the case of the NPSNET-V system [12]:

*The NPSNET-V component framework owes much to JADE, including its concepts of hierarchical containment and the component lifecycle, and much of its terminology.*

The VESLOM Design Methodology and reference architecture, along with some reference implementation of various component layers, have been adopted by undergraduates in computer science from the University College of London as final year project. This allowed them to build smaller components that operated together, contributing to a rich VE as a result.

Acknowledgements

This work was sponsored by the Portuguese Foundation for Science and Technology. The work has benefited from the useful and insightful comments of Anthony Steed, Jon Crowcroft, Mel Slater.
References