

# Mapping Requirements to Architecture: an Experience Report from the VIVIAN Project

**Titos Saridakis**

NOKIA Research Center

PO Box 407, FIN-00180 Helsinki, Finland

Tel: +358 7180 37293 Fax: +358 7180 36308

[titos.saridakis@nokia.com](mailto:titos.saridakis@nokia.com)

## ABSTRACT

The goal of the VIVIAN project (ITEA 99040) is to develop a middleware platform which will promote the development of 3<sup>rd</sup> party, component based, mobile applications. For this purpose a set of some 50 requirements were elicited from the usage scenarios created by the project partners and they were classified into 9 categories according to the system qualities they describe (interoperability, security, network transparency, etc). However, this classification scheme did not provide a clear insight on how to map the collected requirements to the system architecture. This document describes two additional requirement classification schemes, one from the architectural aspects viewpoint which is based on three system models presented in the document and the other from the abstraction levels viewpoint which produces four layers of abstraction. The resulting 3D classification scheme is shown in this document to provide a better understanding of how to map requirements to the VIVIAN architecture. This document also presents the three stages of the iterative process that is followed for the development of the VIVIAN architecture. The effort described in this paper is primarily the architect's task of transforming requirements to system architecture. The contribution of this work is the outline of the process for mapping requirements to architecture which is used for the VIVIAN requirements.

## KEYWORDS

Component-based development, Mobile communication, Requirements engineering, System architecture.

## 1 Introduction

Mobile communication, personal computing and distributed information services are of a global strategic importance in the near future. Commercial PDA platforms providing these services already exist, however their general computational capacity is still rather modest, the ability to connect with remote information services and the Internet are limited, and the number of available software applications for one particular system is very small in comparison, for example, to the open PC/Windows platform. In this context the goal of the VIVIAN project (ITEA<sup>1</sup> 99040) is to develop a middleware platform which will promote the development of 3<sup>rd</sup> party, component based, mobile applications. To reach this objective the VIVIAN project intends to explore technologies like the Symbian OS for PDAs [5] and wireless CORBA [2] in order to provide a platform based on full-fledged component technology, extensible on request according to the application needs. The focus of this paper is placed on the mapping of the VIVIAN requirements to system architecture. The effort described in this paper is primarily the architect's task of transforming requirements to system specification.

This document is structured as follows: §2 describes in brief the VIVIAN background as well as the requirements elicitation results. §3 discusses three system models which introduce a classification scheme for the VIVIAN requirements that helps the mapping of the collected requirements to architectural aspects of the VIVIAN platform. §4 presents another classification scheme for the VIVIAN requirements which organizes them into four layers corresponding to different levels of abstraction. The document concludes in §5 with a brief summary of the work presented in this document and a presentation of the current status of the VIVIAN project.

## 2 Background

The objective of the Vivian project is to provide an adequate platform for handheld terminals and PDA devices (e.g. the PSION netBook and the NOKIA 9210) which will enable the production of 3<sup>rd</sup> party software applications. As a representative use case of VIVIAN the reader can think of the following situation. In the first snapshot of the use case the owner of a VIVIAN-enabled mobile terminal can run client/server based applications over the mobile Internet by contacting remote servers and receiving their replies or by downloading and executing locally software components (eventually agents) that the remote servers send. In the second snapshot of the use case, the owner of the VIVIAN-enabled terminal can interact with other VIVIAN-enabled terminals to run peer-to-peer applications. In the third snapshot, the owner of a VIVIAN-enabled terminal is able to download and execute locally software components (eventually software agents) sent from mobile Internet servers or other mobile terminals. In all snapshots of this general use case, the developers of application components do not have to know the specifics of other application components (e.g. in which language there are coded, what communication API they are using for their interactions, etc) that may potentially interact with their code. Moreover, they can develop their application relying on a set of generic services (e.g. naming service, event handling service, connectivity management service, etc) provided by the VIVIAN platform.

The design of the VIVIAN platform started with each project partner contributing a usage scenario for the VIVIAN platform. Based on the collected set of usage scenarios the requirements elicitation process took place in three steps according to a well established requirements engineering practice [4]. In the first step each partner examined its own scenario and elicited the associated requirements. In the second step, the elicited requirements were assign to a different partner for review. In the third step the reviewed requirements were processed during a VIVIAN workshop in order to eliminate duplicates and to merge those that were considered by the workshop participants to be close enough. Also, the workshop organized the VIVIAN requirements according to a classification scheme which produced 9 different categories for the set of approximately 50 VIVIAN requirements [6].

The basic criterion in the aforementioned classification scheme was the quality of the VIVIAN platform that the requirements were influencing. Hence, the classification included categories regarding interoperability, security, resource management, configuration management, network transparency and others. However, the requirements in each of these categories did not belong to the same level of abstraction. Some are generic requirements that apply in most cases of middleware platforms, some others are specific to mobility and some others are specific to application domains. Moreover, the applied classification scheme did not provide any insight regarding the aspect of the VIVIAN platform (terminal, network or application) to which the requirements were referring to.

## 3 VIVIAN System Models

In order to enable the engineering of the VIVIAN requirements and the development of the system architecture, an additional classification was necessary which would assign requirements to architectural aspects of the system platform. This classification is employed by means of three complementary system models (the network, the terminal and the application models) which present different viewpoints on the architectural aspects of the

---

<sup>1</sup> <http://www.itea-office.org>

VIVIAN platform. The remainder of this section briefly presents these models and describes the classification categories they introduce on the VIVIAN requirements.

### **3.1 The Network Model**

The network model captures the network connectivity and communication aspects of the VIVIAN platform. It perceives the system as a set of nodes interconnected by communications links. The network model does not make any distinction among the nodes regarding their mobility characteristics, i.e. whether they correspond to fixed terminals like Internet servers or to mobile terminals like PDAs. Rather, it focuses on the properties of the communication links and describes the connectivity technologies used by the implementation of those, the transport protocols that can be used on top of them, the possible configurations of the nodes and links that VIVIAN platform should support, the format of the messages exchanged over the communication links, the communication modes (e.g. one way asynchronous, two way synchronous, isochronous, etc) supported by the links, the security, robustness and timeliness properties of the links, etc.

The network model introduces a new category of requirements which is orthogonal to all quality categories mentioned in the VIVIAN requirements document [6]. The category of network requirements contains those requirements that influence the connectivity and communication software of the VIVIAN platform, i.e. the connectivity drivers and the communication protocols installed on the (fixed and mobile) VIVIAN-enabled terminals. The requirements engineering process of the network requirements will result in the specification of the VIVIAN components which implement a network abstraction that provides unified access to a number of different communication networks (GSM/GPRS, Bluetooth, WLAN, etc) and allows the customization of the communication characteristics (communication mode, connection security and robustness, message format, etc).

### **3.2 The Terminal Model**

Similarly to the network model, the terminal model also perceives the system as a set of nodes interconnected by communications links but contrary to the former this model focuses on the properties of the nodes. More precisely, the terminal model describes the properties of the mobile terminals (e.g. PDAs) which are seen as a subset of the properties of the fixed terminals. The fixed terminals, which in the VIVIAN context correspond to Internet servers, are assumed to be always connected to the network (fixed or mobile) and to have unlimited computational, storage and power resources. On the contrary, mobile terminals may or may not be in the coverage of some communication network (primarily mobile) or they may be in the coverage of a given network when they start some application, quit that network and later enter another network while the same application is running. Moreover, mobile terminals have limited computational, storage and power resources which impacts on their capability of running applications. The terminal model takes into consideration these properties of the mobile terminals and describes their accessibility capabilities, their connectivity capabilities, the resources they possess and their management, the terminals interaction capabilities with the user (e.g. screen types, sound support, keyboard/keypad, and voice control), etc.

As expected, the terminal model introduces a requirements category complimentary to the one introduced by the network model. This category includes the VIVIAN requirements that refer to terminal resource management (memory allocation, connectivity means, etc) and terminal interaction characteristics with the user (adaptation of application input/output to the presentation and the user event handling capabilities of the terminal). The requirements engineering process of the terminal requirements will result in the specification of the VIVIAN components which implement the abstraction of the VIVIAN-enabled terminal capable of hosting application components and allowing their interactions with application components that run on other similar terminals. The VIVIAN-enabled terminal abstraction provides unified control to the computational capabilities (memory allocation, process/thread creation, etc) of the terminal and to its UI features (input devices like keypad, pointing device and voice control and output devices like screen, sound system, vibrating battery, etc).

### **3.3 The Application Model**

The application model views the VIVIAN platform as an execution system for component-based applications. The existence of an interconnection network (nodes and communication links) is implicit in this model which only deals with services offered by the platform to control the distribution of application components and the properties of their interactions. In other words, the application model perceives the system as a set of components, some of them being part of the application (e.g. application servers offering Geographic Information Services) and some of them offering enabling services for the communication of the application components (e.g. a catalogue of the available/reachable application services or a service that performs application-transparent switching from a GPRS connection to Bluetooth connection). The application model is aware of component mobility and heterogeneity aspects but it does not have to explicitly deal with the intrinsic of these aspects. Rather it describes the properties of the services that hide the intrinsic of mobility and heterogeneity aspects from the application developer. Such properties include plug-and-play behavior of application components, remote user in-

interface support, common exchange format for application data (e.g. date/time, calendar and location information), etc.

The application model introduces the last requirements category in the architectural aspect related classification scheme. It includes the requirements that refer to application level interoperability (e.g. common exchange format for application data), application component distribution and mobility (e.g. component installation and remote user interface), and application component management (e.g. application service registry, application specific event handling, etc). The requirements engineering process of the terminal requirements will result in the specification of an execution platform abstraction that provides the necessary support for the deployment and the execution of application components developed by 3<sup>rd</sup> party vendors.

## 4 The VIVIAN Architecture

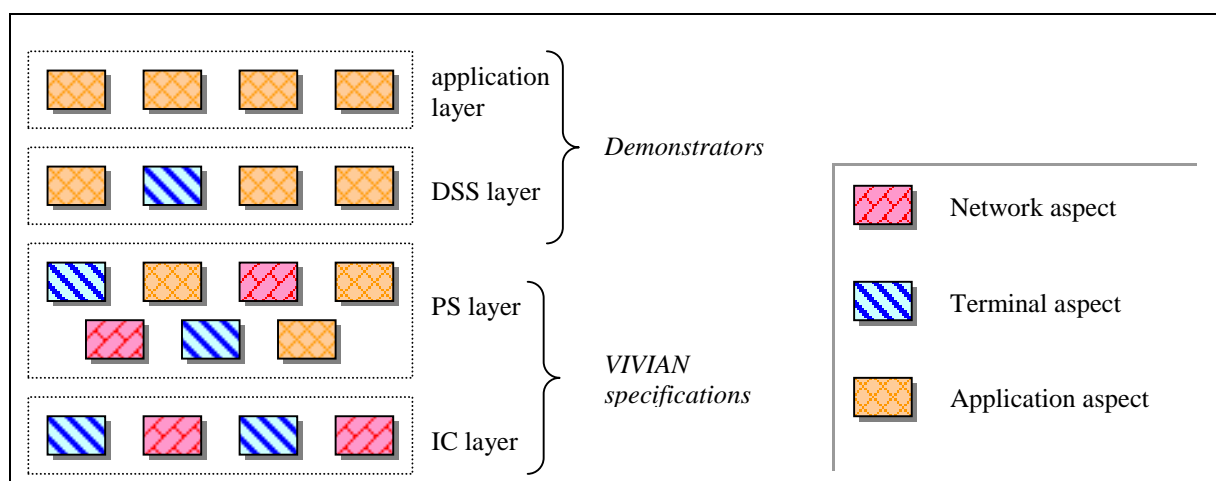
The three system models presented above provide an enhanced understanding regarding the functionalities expected by the VIVIAN platform and outline an initial sketch of the VIVIAN architecture in terms of the network, terminal and application environment aspects. However, the three aforementioned requirement categories contain requirements that belong to different levels of abstraction ranging from connection establishment and process execution to interaction of heterogeneous application components. In order to increase the clarity we have regarding the VIVIAN architecture, we apply the Layers architectural pattern [1] for the decomposition of a system structure into different levels of abstraction and we introduce another classification scheme for the VIVIAN requirements. This section first presents the layered VIVIAN architecture and then describes the incremental process we are following for the development of the VIVIAN architecture.

### 4.1 The VIVIAN layers

When the Layers architectural pattern was applied to the VIVIAN platform and its associated requirements the system was decomposed into a structure of four layers which are described below. The resulting layered architecture of the VIVIAN platform is graphically illustrated in Figure 1.

The lowest layer named *interoperability core* (IC) contains the basic operations regarding the establishment of a network connection and the execution of a software component. It includes functionalities like addressing, messaging, and process/thread creation. These functionalities are very basic and they support all other functionalities provided by the VIVIAN platform. However, being in the bottom of the abstraction level stack implies that using them directly for developing applications is awkward for the developers. Hence, the IC operations are accessible only by the services belonging to the *platform service* (PS) layer which is placed above the IC layer.

Services in the PS layer are further divided into two groups: the *interoperability services* (IS) and the *generic services* (GS). The IS group contains operations that make easier the access to the IC layer and includes functionalities like naming, component registry, event handling, component installation, etc. On the other hand, the GS group contains services which are independent of application domains and which make easier the programming of the VIVIAN platform using the IC and IS functionalities. The functionalities provided by the GS group are persistent storage, atomic transactions, terminal profiling, etc.



**Figure 1** The VIVIAN architecture distinguishing the four abstraction layers and the three architectural aspects.

On top of the PS layer lies the layer of *domain specific services* (DSS) which includes functionalities that are specific to certain application domains, e.g. GIS<sup>2</sup> services, mCRM<sup>3</sup> services, CSCW<sup>4</sup> services, etc. Finally, on

<sup>2</sup> Geographic Information System

<sup>3</sup> mobile Customer Relation Management

top of the DSS layer lies the application layer which contains application components. In general, application components may use more than one DSS layer but the VIVIAN platform does not guarantee the availability of any DSS components. It is left to the responsibility of the application developer to verify that the domain specific services that an application requires exist in a given instantiation of the VIVIAN platform.

#### 4.2 Incremental Stages of VIVIAN Architecture

The last step before starting the specifications of the VIVIAN platform is to organize the system specification process. According to well established software development processes (e.g. the Catalysis approach [3]) the development of a system is an iterative process where a minimal version of the system is developed in the first step, followed by a refinement of the requirements and the system specifications which subsequently leads to the development of a larger version (i.e. with more features) of the system and so on until the entire system is developed. We applied this iterative development process to the VIVIAN platform by identifying a sequence of development stages each of which corresponds to a richer version of the platform compared to the previous stage. Each stage includes parts of all four architectural layers presented in the previous subsection and delivers a subset of the VIVIAN platform which has a restrained set of functionalities compared to the final version of the system.

The initial stage is labeled *Stage 0* and it provides a version of the VIVIAN platform which supports only remote access, i.e. interactions among components distributed on different VIVIAN-enabled terminals. In this stage the IC layer consists of the services that provide addressing and messaging functionalities which includes the mapping of addresses and the message format used by the application components to the addresses and the message format recognized by the actual transport protocol. The PS layer consists of the naming, service discovery and event handling services in the IS group and persistent storage, remote UI and connectivity management services in the GS group. Finally, the DSS and application layers (which are not part of the VIVIAN specification but are used only for demonstration purposes) focus on two application domains: linguistic applications and collaborative drawing applications.

Stage 0 deals only with remote communication of components but does not deal with downloading and installing components on VIVIAN-enabled terminals. This is the focus of *Stage 1* which, in addition to the functionalities provided by Stage 0, provides VIVIAN-enabled terminals with component hosting and execution capabilities. In this stage the IC layer is extended with process and memory management functionalities, the PS layer is extended with component installation, component version control, access control, communication encryption and user authentication services. Finally, the DSS and application layers focus on the application domains of GIS and mCRM.

Building on the version of the VIVIAN platform delivered by Stage 1, Stage 2 will transform the VIVIAN platform into an agent-enabled system. The current understanding of the services and the functionalities provided at the different architectural layers by Stage 2 is not clear yet, but the intention of the project consortium is to have Stage 2 delivering the final version of the VIVIAN platform.

## 5 Conclusion

This document presented in brief the basic steps of the requirements engineering process that has been employed to map the VIVIAN requirements to the system architecture. The initial classification scheme applied on the requirements elicited from the usage scenarios of VIVIAN [6] was based on the system qualities to which the requirements were referring and was proven not sufficient to drive the mapping of requirements to architectural components. Consequently, two alternative classification schemes were introduced in this document, which organized the requirements with respect to the architectural aspect to which they correspond in one case and with respect to the level of abstraction to which they correspond in the other. The resulting 3D classification scheme (qualities - architectural aspects - abstraction levels) provides a clear understanding of the functionalities and services that are described by the VIVIAN requirements and helps their mapping to the structural entities that compose the VIVIAN architecture.

Using the aforementioned 3D classification scheme to lead the exercise of mapping the VIVIAN requirements to the architecture yielded a layered system structure corresponding to four levels of abstraction. Each abstraction level is captured by a layer which contains services that correspond to more than one of the three architectural aspects also identified by the 3D classification scheme. The resulting system architecture is summarized graphically in Figure 1. Finally, the development of the VIVIAN architecture is organized as an iterative process in three stages: the Stage 0 focuses on remote access, Stage 1 focuses on component download and installation and Stage 2 aim at transforming the result of Stage 1 into an agent-enabled system.

Currently, Stage 0 of the VIVIAN architecture is about to be completed and includes the specifications of the services and their APIs, and a prototype implementation of the IC and PS layers based on mobile CORBA [2] is

---

<sup>4</sup> Computer Supported Collaborative Work

expected before the end of the year. Right after the delivery of the Stage 0 specifications, which includes the refinement of VIVIAN requirements as a result of their mapping to system architecture, the specifications of Stage 1 will start. The demonstrators (DSS and application components) for Stage 0 will be delivered in the 1<sup>st</sup> quarter of next year. The VIVIAN project will have accomplished its targets (Stage 1 demonstrators and Stage 2 specifications) by the end of the 3<sup>rd</sup> quarter of next year.

## 6 References

- [1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: Pattern-Oriented Software Architecture: a System of Patterns, Volume 1; John Wiley & Sons, 2001.
- [2] OMG Document telecom/2001-03-01: Wireless Access and Terminal Mobility in CORBA; OMG, 2001. <http://www.omg.org/cgi-bin/doc?telecom/01-03-01>
- [3] D. F. D'Souza, A. C. Wills: Objects, Components, and Frameworks with UML: the Catalysis<sup>SM</sup> Approach; Addison-Wesley, 1998.
- [4] I. Sommerville, P. Sawyer: Requirements Engineering: a Good Practice Guide; John Wiley & Sons, 1997.
- [5] SYMBIAN: Symbian OS; Symbian Ltd., 2000. <http://www.symbian.com>
- [6] VIVIAN Consortium: Requirements Document, v 0.8; May 2001. <http://www-nrc.nokia.com/Vivian>