A Platform for Universal Access to Applications

Nuno Valero Ribeiro[1] and José Manuel Brázio[2]

[1] Escola Superior de Tecnologia de Setúbal, Instituto Politécnico de Setúbal, 2910-761 Setúbal, Portugal

[2] Instituto de Telecomunicaões, IST, 1049-001 Lisboa, Portugal

Abstract. This paper gives an insight on the services that are necessary for a system capable of supporting one practical application of the concept of Ubiquitous Computing. The applied scenario is an academic campus and it is pretended that students may access typical computer applications ubiquitously, i. e., anywhere and using any computer. We call it *Universal Access to Applications*. In this scenario, each user may access and use an arbitrary, and heterogeneous, set of applications on any computer and anywhere in the campus. A survey on technologic solutions for enabling the access to non-native applications is firstly summarized. Then we proceed with the design of such distributed system using MoNet methodology. Four steps are covered together with their main contributions: requirements capturing, design of a Logical Model, elaboration of a Functional Model, and finally, setting of a Reference Model for implementation. The proof-of-concept platform, developed on basis of the designed system, proved the concept. Finally, conclusions about the work done and the concept future applications are considered.

1 Introduction

In the last decades we have been witnessing a tremendous evolution in the computer hardware and software industry. Today we use a variety of personal computational devices, ranging from desktop to mobile PC's and PDA's. Although the computational autonomy, and hardware diversity, provided for the user has increased, still, one can not *arbitrarily* use any of these devices for one's computational needs: the desired user applications and environmental settings may not be found in one particular device the user may actually be using in one particular circumstance.

We considered this problem in the campus of an academic institution. We want to enable the access to the execution of arbitrary computer applications to users. Independently of the computational device they may use in a particular moment. Without requiring additional effort to the user. The computer applications are made available from a distributed environment by a number of application servers.

For this purpose the first technical problem to be solved stems from the heterogeneity of computational platforms vs. nature of computer applications that may be found on an academic campus. Namely, the problem is: how to give access to the execution of applications that were compiled for other platforms. Another problem is the definition of a distributed system for the support of interaction between application clients and users. This results in a typical Distributed System modelling problem and involves the support and integration of aspects such as: data distribution and transparent access, security aspects, fault tolerance, and name and location management. These problems are well identified and studied in proper literature [1]. Other issues arise due to the possible use of *mobile* client platforms. They concern communication problems such as, network location management or data synchronization, and have been subject of study in the area of Mobile Computing [2][3].

In this paper a distributed system for an academic campus environment will be studied. In section 2 the issue of non-native application execution is examined, for which a number of technological alternatives are listed and briefly assessed. In section 3 the architectural design of the system is presented, together with a brief description of MoNet design methodology, followed by the results of its design steps. Section 4 shortly describes the developed proof-of-concept platform and subsequent lessons learned. Finally, section 5 enumerates major conclusions.

2 Using non-native applications

A survey made on the technological solutions for the execution of non-native applications shows that these solutions fit in four different main approaches (more details in chapter 3 of [4]):

1. *Exporting the application user interface*—which assumes running the application on its native platform and redirecting its user interface to the client, either *directly* using built-in windows systems mechanisms, or *indirectly* using proxy entities;
2. *Running platform independent applications*—developing applications in programming languages that generate platform-independent code;
3. *Platform emulation*—running non-native applications on a emulator of the native platform;
4. *Process migration*—migrating processes via specific operating system mechanisms allowing application processes to move and run on other platforms.

Direct mechanisms for exporting the application user interface inherently restrict themselves to a family of operating systems (and sometimes not even the entire family). In order to support different families of operating systems *indirect* exporting mechanisms ought to be adopted. Java programming language has been widely accepted for the development of platform-independent code. Of course, but unfortunately, Java does not easily integrate software that is already compiled for other platforms. Emulation technology lacks the efficiency and performance of the natively running application. Plus, most of times, it is not technically possible to emulate every feature of the native platform. Migration of application processes, between different operating system families, is out of the question since it raises extremely complex problems due to operating systems software complexity and architectural differences.

This leaded us to elect the indirect way of exporting the application user interface to the client device as the most suitable technique for universal access to heterogeneous computer applications.

3 Designing the system

For the design phase of the distributed system we adopted MoNet methodology [5], that comprises four main steps: (1) identification of requirements and services; (2) design of a Logical Model, which identifies the system main function modules; (3) construction of a Functional Model, defining its Functional Entities, their relationships and information flow; and, (4) elaboration of a Reference Model, which takes into account implementation aspects.

In the following subsections we summarize the system design main results (from hereon also referred as SDUA, standing for *System for Universal Disponibilization of Applications*).

3.1 Requirements and services

Briefly, we have identified the following requirements for each kind of user:

1. *end user*:
   (a) universal application access — ability to access applications independently of its physical location and terminal computer;
   (b) access to the execution of heterogeneous applications — ability to use applications, which may be different in their nature and characteristics, independently of the device being used;
   (c) support for access to a private data storage system — ability to access a private file system, available for any device that may be used;
   (d) session control — ability to start and terminate a session whenever the user wishes;
2. *manager*: capability to manage the system, creating users, setting their profiles, adding/removing applications, monitoring operation, verifying correct component work, and auditing behavior to detect and trace wrong procedures or security attacks;

On the basis of the requirements, four SDUA main services were identified, as illustrated in figure 1:

1. Universal Access to Applications
2. Access to Private Data
3. SDUA Management
4. Computational Services Registry

3.2 Logical Model

The Logical Model is the collection of the business logic modules identified for each one of the main services. Each of these services is supported by a set of functions, that are initially identified and further refined, in an iterative top-down approach, until the elementary functions are met. Then, these are grouped in Logical Entities, when related to each other by one determined functional criteria. If the same Logical Entity is found

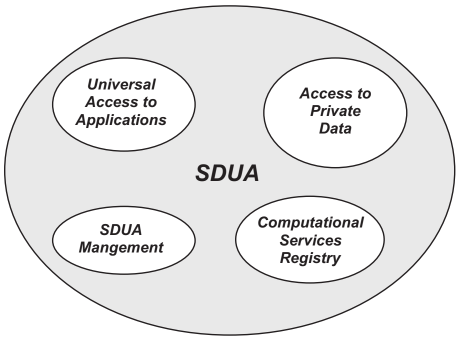
SDUA
Universal Access to Applications
Access to Private Data
SDUA Mangement
Computational Services Registry

Fig. 1. SDUA main services

in more than one Logical Model (there is one Logical Model for each main service), it is "promoted" generating one hierarchically superior Logical Entity.

As an example, figure 2 illustrates the two refinement steps taken for achieving the Logical Model of the *Universal Access to Applications* service. Its Logical Entities are fully described, enumerating its functions, textual description of interfaces and behavior, in [4].

3.3 Functional Model

The Functional Model decomposes, further on, the system into Functional Entities and their relationships. It is derived from the Logical Model by taking distribution aspects into account like physical machine allocation of functions.

Four different physical tiers were identified:

1. the *client*: any computational device used for accessing the execution of applications;
2. the *application server*: set of computers that offer the possibility of hosting the applications execution;
3. the *SDUA server*: set of computers where the SDUA's main services are allocated;
4. the *Data server*: set of computers, and their respective operating file systems, for archiving user data.

Logical Entities may now be divided in disjoint sets of functions according to the physical tier that hosts them. Each of these sets of functions represent a Functional Entity. Additional Information Flow diagrams specify the interaction between these Functional Entities ([3]).

3.4 Reference Model

The Reference Model defines groups of Functional Entities (*Functional Groups*) taking into account criteria related to implementation aspects. These criteria are elected con-

[3] Complete descriptions and diagrams may be found in [4]

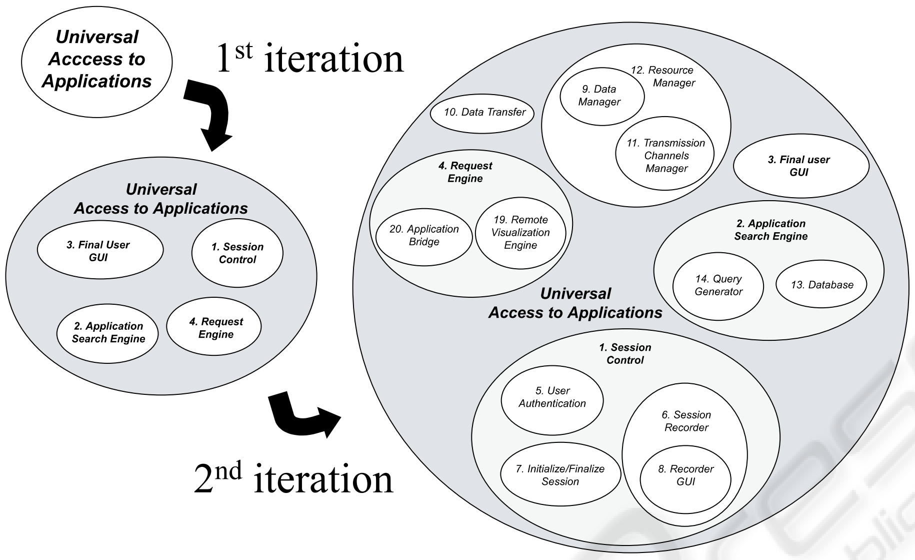

Fig. 2. Refinement example of Universal Access to Applications Logical Model

sidering the goal of the analysis that may be done after reaching the Reference Model. It may take into account either technological or economical aspects.

Two technological criteria were considered. One takes into account the distribution of Functional Entities among tiers (already identified in 3.3, namely: *client*, *application server*, *SDUA server*, and *Data server*). The other one concerns the logical distribution of these Functional Entities. This logical distribution adopted an architectural perspective of the Internet proposed by Miroslav Benda in [6]. It divides the system into five logical parts: user interface, data, business logic, delivery system, and middleware (which "glues" all previous four).

From the analysis of the obtained Reference Model, after applying the two technological criteria previously exposed, we may state that:

1. the largest number of Functional Entities are hosted in *SDUA's server* Functional Group;
2. the most representative logic part is the *middleware*.

We may now state that:

1. the hardware and software platform chosen for hosting *SDUA's server* Functional Entities is fundamental for the overall system performance;
2. a careful choice of the adopted middleware is essential for development and deployment of the system since it strongly affects the functioning of the whole.

4 Proof-of-concept platform

A proof-of-concept platform, based on SDUA, was developed for testing the concept of Universal Access to Applications on an academic campuses. This platform offers its users, on either Windows or Linux based client platforms, the use of a Microsoft application or a X windows based application hosted on applications servers.

The implementation of this proof-of-concept platform required the development of some key modules, from those already identified for the global system, namely:

– Initialise/Finalise Session, for being able to start and end a session;
– End User Interface, for being able to interact with the system;
– Request Engine for handling requests, Remote Visualization Engine for accessing applications running on their servers;
– System's Database, for archiving information about users.

We based the Remote Visualization Engine on VNC technology [7] as previously supported in section 2. VNC system is based on a remote video frame buffer protocol and, therefore, can be used with any operating system family.

During the development phase UML was used for object-oriented modelling. The correspondence between MoNet and UML models was readily established by means of:

– each Logical Entity defined one, or more, object classes, and the relations between these Entities defined their class associations;
– hierarchical groups of Logical Entities were set via packages;
– Information Flow diagrams, specified with the system's Functional Model, identified method calls among objects;
– Functional Models also clarified how objects were physically distributed among different computers.

The deployed platform was tested and worked properly. End users were able to use either a MS application or a X Windows application, as intended, simply by requesting their remote execution through a basic system menu.

Next we intend to improve the developed platform by allowing the addition and removal of applications in a dynamic fashion. We are considering applying Jini [8] technology as an infra-structure for registering/unregistering applications taken as incoming and outgoing services. Another improvement is being considered by means of adding functionalities for load balancing the cluster of *application servers* regarding performance aspects.

5 Conclusions

We have considered the problem of using heterogeneous applications at an arbitrary computational device. We initially surveyed the technological solutions for accessing the execution of non-native applications. For the details of architectural development, a system for an academic campus was considered. In the design phase of the system, we concluded that middleware is a key factor for the development and deployment of such a system, and, hardware and software for hosting determined functionalities are fundamental for the overall system performance. We also enunciated some bridge points between MoNet's and UML design methodologies. The developed proof-of-concept platform used, as a solution for remotely access non-native applications, indirect mechanisms for exporting the user interface of the application to the end user device. It has demonstrated the feasibility of the system.

As explained in [9], there is room in a campus environment for the advantageous use of such system. For the same reasons, among others, certain commercial organizations may also benefit from such a computer system approach. A lease scheme of offering computational application services on demand may be a possibility. Under this scenario the user could get instantaneous and transparent access to any bug fixes or upgrades to applications. New services and application providers will have a potential market to explore.

References

1. Coulouris, G., Dollimore, J., Kindberg, T.: Distributed Systems: Concepts and Design. Second edn. Addison-Wesley (1994)
2. Rutgers, T.I., Korth, H.F.: Mobile Computing. Volume 353 of The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers (1996)
3. Milojicic, D.S., Douglis, F., Wheeler, R.: Mobility : Processes, Computers, and Agents. Addison-Wesley Publishing Company (1999)
4. Ribeiro, N.V.: Uma Plataforma para Acesso Universal a Aplicações. Master's thesis, Instituto Superior Técnico, Lisboa, Portugal (2000) URL, http://ltodi.est.ips.pt/nribeiro/#MSc, accessed July 2000.
5. Katoen, J.P.: The MoNet Design Methodology. MoNet deliverable Technical Draft 3.1 MoNet/GA3/UT/006, University of Twente (1993)
6. Benda, M.: The Architecture of Global Access. IEEE Internet Computing **1** (1997) pp. 78–80
7. : VNC — Virtual Network Computing (1999) ©AT&T Laboratories Cambridge, URL, http://www.uk.research.att.com/vnc/, accessed April 2000.
8. Sun Microsystems, Inc.: Jini^{TM} Architectural Overview. (1999) URL, http://www.sun.com/jini/specs, accessed July 2000.
9. Ribeiro, N.V., Brázio, J.M.: Campus Personal Computing: uses, evolution, and new perspectives. In: Actas da III Conferência de Telecomunicações, IT (2001) 315–318