Keeping Emulation Environments Portable

FP7-ICT-231954

Guideline document and peripheral input/output libraries for developments

Deliverable number	*D 5.1*
Nature	*Other*
Dissemination level	*PU*
Delivery date	**Due: M8** (September 2009) **Actual: M8** (September 2009)
Status	*Final*
Workpackage number	*WP5*
Lead beneficiary	*UPHEC*
Author(s)	*Antonio Ciuffreda, UPHEC* *David Anderson, UPHEC* *Janet Delve, UPHEC* *Getaneh Agegn Alemu, UPHEC* *Dan Pinchbeck, UPHEC* *Bram Lohman,TSSP* *David Michel, TSSP* *Bart Kiers, KB* *Vincent Joguin, JOG*

Document history

Revisions

Version	Date	Author	Changes
0.1	3 September 2009	Bram Lohman	Sections 3.2 and 3.3
0.2	5 September 2009	Vincent Joguin	Section 3.3
0.3	11 September 2009	David Michel	Sections 1.2, 2.1, 2.2, 2.3, 3.1.1, 3.1.2, 3.1.4, 3.2, 3.3, 4.2.1 and 4.2.7
0.4	15 September 2009	Vincent Joguin	Executive Summary, Abbreviations, Sections 3.1.1 and 3.1.4
0.5	15 September 2009	David Anderson	Executive Summary, Abbreviations, Sections 1, 2, 3 and 4

Reviews

Date	Name	Result
23/09/2009	Jean Marc Saffroy	Approved with changes
29/09/2009	Jeffrey van der Hoeven	Approved with changes

Signature/Approval

Approved by (signature)	Date

Accepted by at European Commission (signature)	Date

Executive Summary

This report presents the initial integration strategy of the KEEP Emulation Access Platform (EAP) components, with a focus on the methodologies used to integrate the Core Emulation Framework (CEF) with the Olonys Virtual Machine (VM), the Front-end Emulation Framework (FEF) and external web services. In this context the technical choices made such as the use of the compiler from the GNU Compiler Collection (GCC), the adoption of Java for the development of the EF components, the consequent choice of the IcedTea6 Java Virtual Machine (JVM) and the rejection of any restriction of usage of the peripheral input/output libraries are explained. Also the report offers a set of lists of conventional programming styles and software technologies for the design and development of the CEF and FEF.

Abbreviations

Advanced Interactive eXecutive	AIX
Application Programming Interface	API
Abstract Window Toolkit	AWT
Bibliothèque nationale de France	BnF
Commodore 64	C64
Connected Limited Device Configuration	CLDC
Core Emulation Framework	CEF
Deutsche Nationalbibliothek	DNB
Emulation Access Platform	EAP
Emulation Framework	EF
Front-end Emulation Framework	FEF
GNU Compiler Collection (formerly the “GNU C Compiler”)	GCC
GNU’s Not Unix	GNU
General Public License	GPL
Graphical User Interface	GUI
Hyper Text Markup Language	HTML
Input/Output	I/O
Integrated Development Environment	IDE
Java Development Kit	JDK
Joguin SAS	JOG
Java Virtual Machine	JVM
Joint Photographic Experts Group	JPEG

Koninklijke Bibliotheek	KB
Keeping Emulation Environments Portable	KEEP
Kooperativer Aufbau eines Langzeitarchivs	KOPAL
Multiple Arcade Machine Emulator	MAME
Multi Emulator Super System	MESS
Operating System	OS
Object Oriented	OO
Personal Computer	PC
Portable Document Format	PDF
Remote Procedure Calls	RPC
Simple Object Access Protocol	SOAP
Système de Préservation et d'Archive Rèparti	SPAR
Scalable Processor ARChitecture	SPARC
Scalable Vector Graphics	SVG
Subversion (version control system)	SVN
Technology Compatibility Kit	TCK
Tessella plc	TSSP
Transfer Tool Framework	TTF
Unified Modeling Language	UML
Unified Digital Formats Registry	UDFR
University of Portsmouth	UPHEC
Virtual Machine	VM
Work Package	WP
Web Service Description Language	WSDL

Table of Contents

List of Figures

1 Introduction

1.1 Objectives and Scope

The main aim of this report is to provide a clear insight into the strategy adopted for the integration of the different components of the Emulation Access Platform (EAP). This includes the core and front-end of the Emulation Framework (EF) as well as the Olonys Virtual Machine (VM). Furthermore, the Transfer Tool Framework (TTF) is briefly described although, due to its detached position within the EAP, an extensive integration with the other components is not required. The report also seeks to offer a reference guide to the conventional technologies and coding standards which will be used in the design and development process of these components.

1.2 Outline of the Document

Section 2 provides a brief description of the capabilities of the TTF, and the contextual role of ech of the main components of the EAP:

- the Olonys VM
- the EF, which in turn is comprised of:
    - the Core Emulation Framework (CEF)
    - the Front-end Emulation Framework (FEF),

Section 3 offers a clear description of the methodology adopted for porting the CEF on top of the Olonys VM. Guidelines for the integration of the CEF with the FEF and with external services are discussed.

Also discussed are

- issues regarding the choice of the programming language with the potential restrictions usage of peripheral input/output libraries
- the choice of the JVM
- the choice of compiler.

Section 4 sets out the list of conventions regarding the coding style and the different tools used for the design and development of the CEF and the FEF.

2 General Capabilities and Context of the Emulation Access Platform Components

The EAP consists of various components:

- Transfer Tool Framework (TTF).
- Core Emulation Framework (CEF).
- Front-end Emulation Framework (FEF).
- Olonys Virtual Machine (Olonys VM).

Figure 1 depicts these components on a general level.

2.1 Transfer Tool Framework

The Transfer Tool Framework (TTF) offers data managers the facility to easily search and select appropriate tools for the migration of digital data from old data carriers to current storage devices in the form of usable content (disk images). The TTF operates independently from other EAP components, as the activities performed by this component are related to the ingestion process of the migrated digital data into the Digital Archive; tasks performed by other components instead are associated with the retrieval process of this data. Nonetheless, the way digital data is captured from the old data carriers and the metadata used to describe it influences enormously the capabilities of the EF in the object rendering activity. Therefore, the TTF will be developed in close cooperation with the EF thus ensuring that the metadata model and the formats of the data object meet the functionality of the EF. This however does not imply a real integration between this component and the EF.

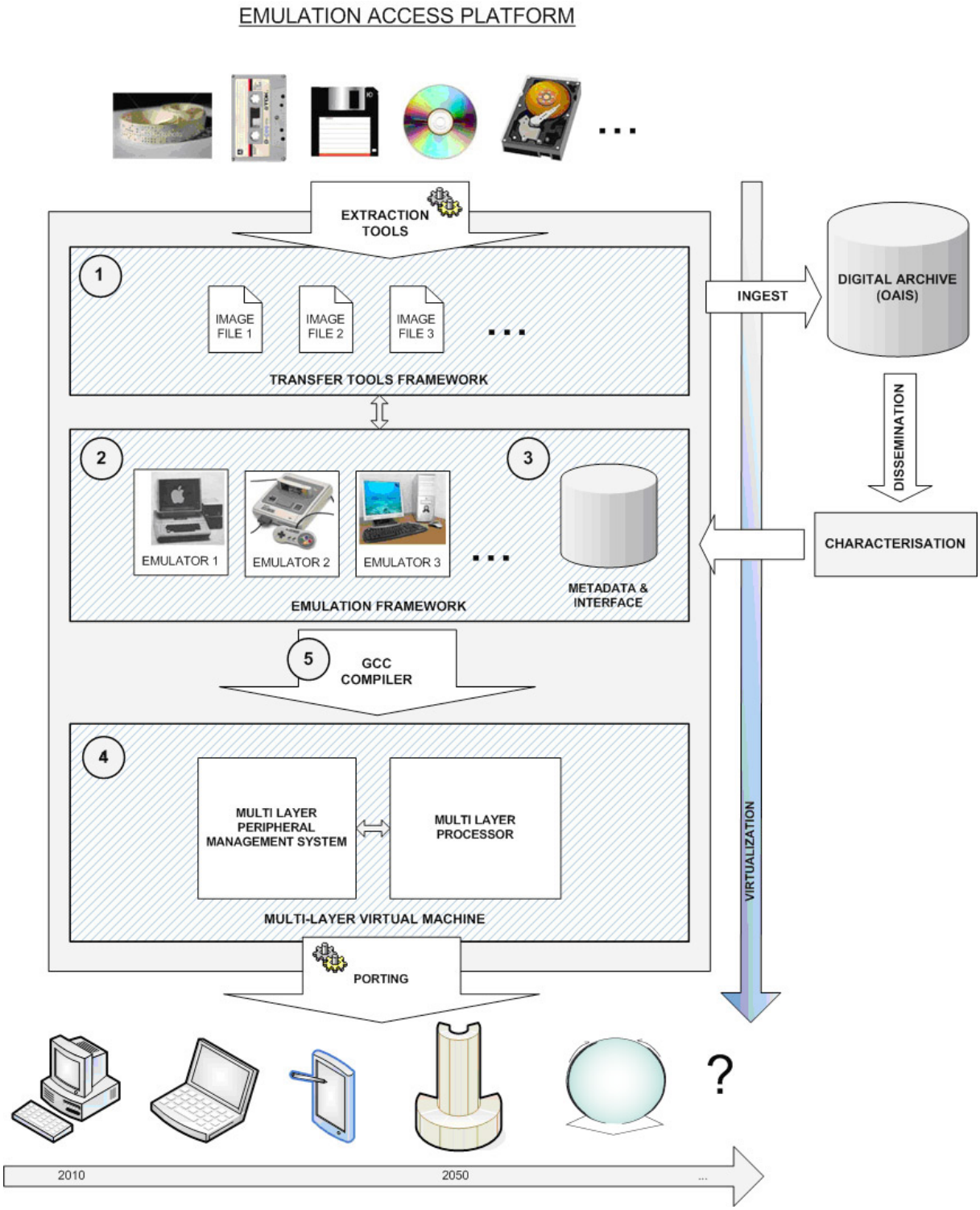

Figure 1: An overview of the Emulation Access Platform (Section 2).

2.2 Core Emulation Framework

The Core Emulation Framework (CEF) is instrumental in the presentation of digital objects stored in external library catalogues (E-Depot[1], KOPAL[2], etc.) in their original hardware and software environment. The CEF (see Figure 2) makes use of external technical registries, such as PRONOM[3] [4],

to characterise a digital object from a library institution database. Successively the characterisation data and/or the pathway (describing all software and hardware dependencies of the digital object) issued by the external technical registry are then used by the CEF to select from an internal archive the necessary emulator and other software (operating system, applications, drivers, etc.). These are then used by the CEF to present the digital object through the FEF.

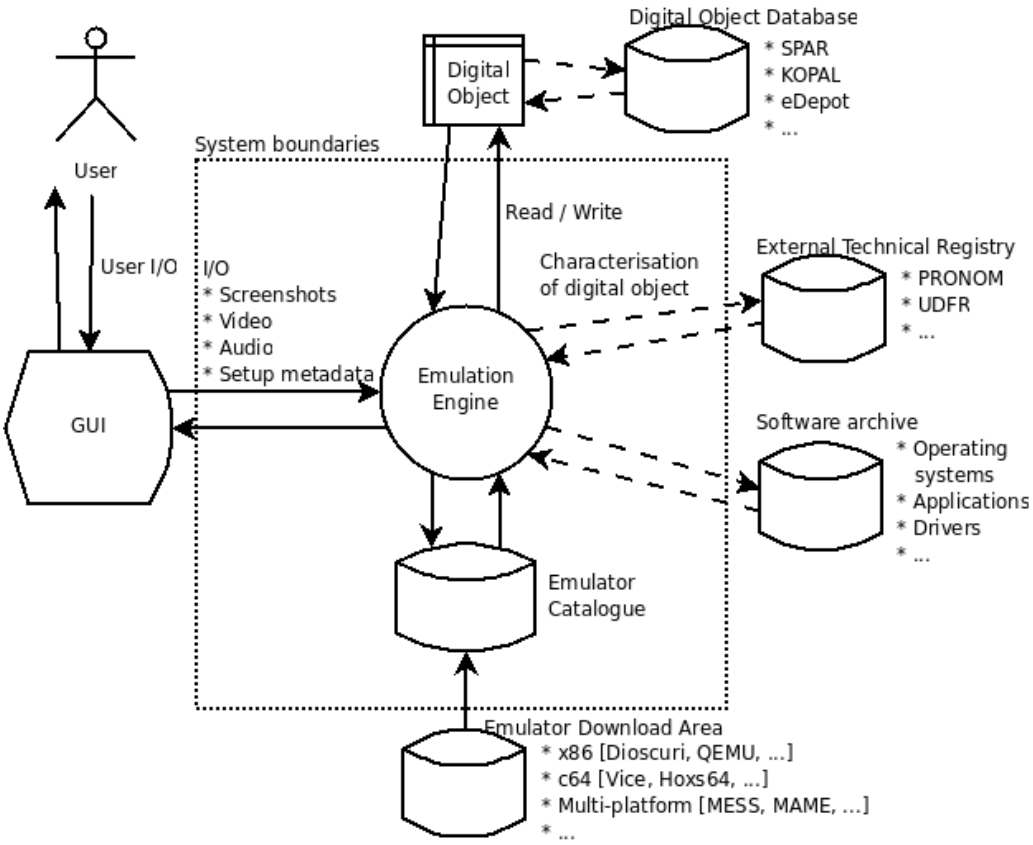

Figure 2. A first design showing the structure of the Core Emulation Framework and the components interacting with it, such as the Front-end Emulation Framework (GUI), external Technical Registries and Library Databases (Section 2.2).

2.3 Front-end Emulation Framework

A Front-end Emulation Framework (FEF), also known as the Graphical User Interface (GUI) of the EF, usually provides users with an interface through which they can access and interact with the specific digital object in their emulated original environment. As previously stated, the emulated environment is chosen by the CEF which uses external technical registries to characterise the object, obtain the related emulation pathway and pass it to the FEF.

2.4 Olonys Virtual Machine

In order to be portable to different computer architectures and durable in a long term, the emulation processes of the EF will not run directly on the host machine. It will run instead on the Olonys VM (see Section 3.1), a virtual machine which was designed specifically to ensure portability and flexibility. This is achieved in the Olonys VM via a multi-layered software structure and a specific self-adaptive peripheral management system. This enables any application, including the components of the EF, intended to run in the Olonys VM to be compiled in a virtual machine system which is almost identical to a real computer system. In addition no complex architecture is needed for the Olonys VM to be ported to a real computer.

3 Integration Strategy of the Emulation Access Platform Components

3.1 Integration of the Core Emulation Framework with the Olonys Virtual Machine

In order to ensure sustainability, the EF should run on any computer platform now and in the future. Although this represents a difficult challenge, the use of the Olonys VM, designed to offer a high level of portability, may be the long term solution. In order to run on top of the Olonys VM, the EF (in particular the CEF) should be compatible with the Olonys VM architecture and instruction set. The development of a full-fledged version of the EF which complies with to the Olonys VM specifications is a complex and lengthy task. As result, a transitional phase with an initial integration approach has been adopted.

3.1.1 Initial Integration Strategy

The initial strategy for the integration of the EF with the Olonys VM (see Figure 3) aims to deliver a first prototype which can demonstrate the ability of these two components to operate jointly without providing a limit to the capabilities of the EF. This approach should overcome any usage restriction of peripheral input/output libraries by the EF components.

The first prototype will demonstrate the successful execution of at least two existing emulators (see internal deliverable 2.1). The first emulator will emulate the environment of a C64[5] computer architecture. This emulator will execute directly on top of the Olonys VM due to its compilation via the C Compiler (GCC)[6] into the Olonys VM. The second emulator will emulate the environment of a PC/x86[7] computer architecture which will execute via a Java Virtual Machine[8] (JVM) on the host machine. The use of a JVM is regarded as a transitional solution for portability. Although the level of portability of the JVM is inferior in comparison with the Olonys VM, the JVM can run on a wide range of existing computer platforms (Windows[9], Linux[10] [11], MacOS[12], Sparc[13] and AIX[14]). Both emulators will be controlled by the EF.

In order to implement the aforementioned integration strategy three important choices need to be made:

- The main programming language of the EF.
- The Java Virtual Machine for supporting the Java-based X86 emulator.
- The compiler for translating the C64 emulator code into Olonys VM compliant code.

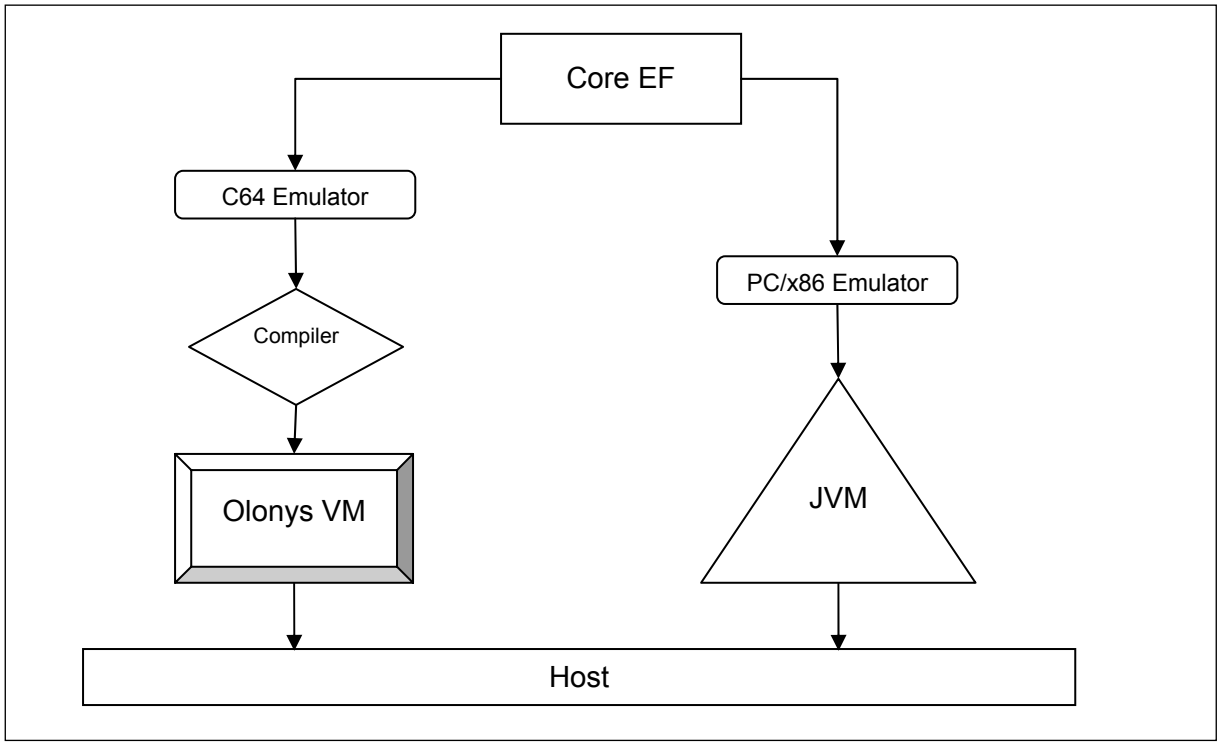

Figure 3. A diagram showing the initial strategy for the integration of the Emulation Framework with the Olonys VM (Section 3.1.1).

3.1.2 The Programming Language and the Use of Peripheral Input/Output Libraries

Two Object-Oriented languages, C++[15] and Java[16] were considered for the development of the CEF. Both languages are robust, extensible and support code re-use due to their Object-Oriented[17] (OO) nature. There is clearly a balance to be struck: Java provides more libraries than C++ but has slower execution speed. An important consideration in favour of Java is the fact that it was used for the development of important external web services (e.g. PRONOM) which are to be used in the CEF. Finally, the EF development team already has considerable experience working in Java.

As result of the adoption of Java for the development of the CEF, the use of a restricted number of specific peripheral input/output libraries, previously considered to facilitate the porting of the EF to the Olonys VM, has become redundant. Consequentially it was decided that no constraints were given in this context for the use of any peripheral input/output library.

3.1.3 The Java Virtual Machine

Following the adoption of a specific programming style (OO) and language (Java), the choice of an appropriate JVM is needed. In addition to OpenJDK[18], Sun Microsystems' own freely available but partly closed-source JVM, various alternatives exist. An important requirement to consider for the choice of the future JVM is its open source nature. Being able to access the source code of a JVM will allow the developers of the Olonys VM to port the JVM to this VM. Although this process is likely to require a considerable amount of effort and may, ultimately, not be feasible within the scope of the KEEP project, it remains nevertheless a valid solution.

Several existing open-source JVMs have been investigated for the execution of the EF under the Linux OS. A number of issues were encountered:: Kaffe[19] and SableVM[20] have not been updated for a considerable amount of time; Squawk[21] was designed for Connected Limited Device Configuration (CLDC)[22]-based applications and was targeted mainly for small devices; CACAO[23] and JamVM[24] rely only on the GNU Classpath[25], which is incomplete, as class library.

In the end, there were only two viable alternatives: IcedTea6[26] (a packaged version of OpenJDK6) and Apache Harmony[27]. The stable version of IcedTea6 concerns the Java 6[28] implementation with current development being focused on Java 7[29]. IcedTea6 is fully open as it uses the GNU Classpath project to replace only the non-GPL[30] components of OpenJDK. As IcedTea6 utilises the C++-based Sun HotSpot[31] as its virtual machine, a compiler for this language must eventually be created in order to port IcedTea6 on top of the Olonys VM. The main reasons behind the choice of IcedTea6 are the following:

- Good coverage of the class library.
- Approval from the Technical Compatibility Kit (TCK)[32] from Sun.
- Compatibility with web services libraries such as Xfire[33] or CXF[34].
- Compatibility with Java GUI libraries such as Abstract Window Toolkit (AWT)[35] and Swing[36] and other commonly used Java libraries.
- Large amount of available documentation.
- Large user community.

In comparison with IcedTea6, Apache Harmony offers less extensive class library coverage. This coverage deficiency can cause major issues when certain non-trivial libraries, such as the ones for web services, are used. Additionally Apache Harmony complies only with Java 5 (development for Java6 is still in progress) and provides less documentation than does IcedTea6.

3.1.4 The Compiler

In order to enable existing applications to run directly on the Olonys VM a compiler that translates the source code of the native application into Olonys VM compliant code is needed. Therefore, a C compiler (GCC)[37] which transforms the source code of a C-based application into pure machine code that the Olonys VM can understand and execute will be used. The C language[38] was initially chosen for the following reasons:

- Due to its smaller standard library and simple structure, C requires relatively little development effort by comparison with other languages such as C++ or Java.
- C enjoys a very wide acceptance and most of the existing emulators are written in C. This makes a C compiler a mandatory requirement for the Olonys VM.
- Many common 3rd-party applications, libraries and other programming languages (including the higher-level object-oriented languages previously mentioned) are implemented in C.

The Linux Operating System (OS) will be used as an intermediate interface for the execution of non C-based applications in the Olonys VM (see Figure 4). The Linux kernel[39], being written in C, can be compiled directly for the Olonys VM, thus enhancing the capacity of the Olonys VM to support directly or indirectly the execution of other applications written in various higher level languages such as Java or

C++. Future prototypes of the EF could run directly on the Olonys VM (as depicted in Figure 4). This depends however on the progress of the development of a Java compiler for the Olonys VM.

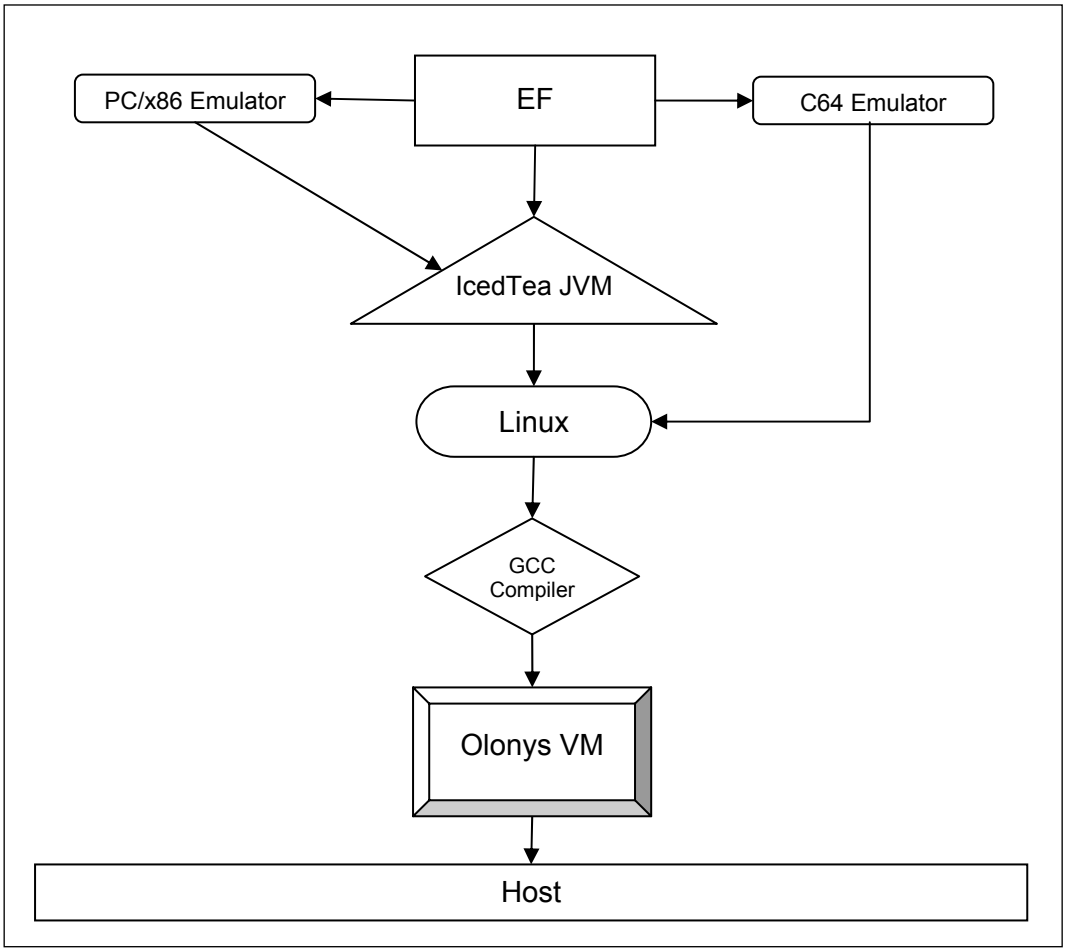

Figure 4. A diagram showing the longer-term integration approach of the Emulation Framework with the Olonys VM (Section 3.1.4).

3.2 Integration of the Core and Front-end Emulation Framework

Due to the common programming language used for the development of the FEF and the CEF, four approaches have been considered for the integration of these two components.

- Use of shared/dynamic libraries[40]. The main benefit of this approach can be found in the possible complete integration between the CEF and the FEF, thus enabling the FEF to access directly the public classes/methods/variables of the CEF. A common language in the two components is required however when using this approach.
- Use of Remote Procedure Calls (RPC)[41]. This approach, although it provides more flexibility and doesn't require the use of a common language for both components, remains difficult to implement.
- Use of the configuration text file to set all possible options. This approach is not very flexible and does not permit any dynamic exchange of information, as only information from the FEF to

the CEF can be sent. In addition the CEF cannot receive more information once it has been launched.

- Issue of commands by the FEF to the CEF through a command line interface. Similarly to the use of the configuration text file this approach is limited as it would be more problematic to send information back to the FEF.

Of these four methodologies the use of shared/dynamic libraries is felt to be the most viable. However depending on the final installation layout, other approaches may be used; i.e. if the EF is delivered as a network application RPC may be used for the communication between components. A definitive decision on the integration of front-end and core frameworks will be taken once the design process of the EF will come to an end.

3.3 Integration of the Core Emulation Framework with External Web Services

The EF is expected to contact several external systems for information gathering: a Digital Object Database, various Technical Registries, a Software Archive and an Emulator Download Area.

The external systems are expected to provide a defined interface, and the Emulation Framework will implement this interface. The communication with these systems is planned to be done via Simple Object Access Protocol (SOAP)[42]/ Web Service Definition Language (WSDL)[43].

The use of web services provides a clear separation for communication between these systems. This means the integration of the EF with external systems is kept to a minimum, provided the interfaces are clearly defined.

4 Development Conventions

4.1 Code Conventions

In order to improve the readability of the software and to decrease maintenance costs, the code conventions established by Sun Microsystems[44] have been adopted. In addition to these code standards, an additional set of rules specific to the KEEP project have been proposed.

4.2 Application Tools Conventions

4.2.1 Integrated Development Environment

Either Eclipse[45] or NetBeans[46] will be used as the Integrated Development Environment (IDE)[47] for the development of the CEF and the FEF. These two IDEs are the only free and open-source development framework for Java development and possess a large support from the developers' community. The large number of features and plug-ins existing in both IDEs guarantees a powerful platform for the development of the EF components.

4.2.2 Designing (UML Diagrams)

Three Java-based non-commercial Unified Modeling Language (UML)[48] editors have been considered for software design activities: Violet[49], UMLet[50] and ArgoUML[51]. Violet is an easy to use cross-platform UML Editor with an intuitive GUI. However it doesn't provide important features such as the ability to design deployment diagrams, code generation, reverse engineering, undo/redo facilities and multiple selections. Similarly to Violet, UMLet offers a user friendly GUI aimed for creating quickly UML diagrams which can be exported in different formats such as PDF, JPG and SVG. However code generation and reverse engineering are also not supported in this design tool. ArgoUML instead provides additional features in comparison with the previous two tools, such as multiple views and the aforementioned reverse engineering and source code generation from diagrams. It offers however a less user friendly GUI and not fully working undo/redo support. Problems in the design of sequence diagram were also encountered.

4.2.3 Building/Integrating

Three build automation tools for Java applications have been taken in consideration: Apache Ant[52], Apache Maven[53] and GNU Make[54]. Apache Ant and Apache Maven are the most widely used building tools for Java applications and have been made appositely for this purpose. In addition they are adopted in the Eclipse and NetBeans IDEs. GNU Make instead remains the most used building tool and remains the default in Linux, although it is mostly used for building C/C++ applications.

4.2.4 Logging

Two logging services have been evaluated for possible use: Apache Log4j[55] and Java's own build logging service[56]. In comparison with the Java native logging package, Log4j provides a larger number of features, such as the automatic log file rotation. Moreover it is undoubtedly the most popular Java logging framework and as result it has have a large support from the developers' community. The

advantage of using the Java native logging package instead is the avoidance of importing external 3rd party libraries.

4.2.5 Testing

The JUnit[57] testing framework has been adopted for testing Java source code. Being a very popular unit testing framework[58] in the Java developer community and very familiar among the KEEP programmers, the choice of this testing framework was straight forward.

4.2.6 Documentation

Javadoc[59] has been adopted in order to generate Application Programming Interface[60] (API) documentation in Hyper Text Mark-up Language (HTML)[61] format from Java source code. In addition to be free and widely used in the Java developers' community, the format used by this documentation generator is used as industry standard for the documentation of Java classes.

4.2.7 Code Repository

A Subversion (SVN)[62] repository will be held at TSSP for the duration of the project (3 years). It is expected that the repository will be migrated to an open-source hosting service such as SourceForge.net or others afterwards. It is expected the use of this open source revision control system will be very beneficial to the software development team for the management of the source code of the CEF and the FEF. The main feature of a SVN repository is its three dimensional nature of its file-system: in addition to the two dimensions of a normal directory tree, a third dimension for the revisions of data is included. This enables users to save not only the new version of a desired file but also the history of all the changes made to that specific file. As result users can retrieve older versions of a file and analyse its modification history. Another important feature of SVN is the ability from different users to edit files stored in the repository concurrently: if the SVN receives two or more edited versions of the same file from two different users, it will merge the changes into one newly edited copy of the file, or it will report conflicts.

Conclusions

The initial strategy of the integration of the KEEP Emulation Access Platform components was presented in this report. The document described the approaches used for the integration of the CEF with the Olonys VM, the FEF and external registries. In this perspective the use of the open source GCC compiler was decided in order to port C-written emulators and non C-written emulators (these will use the Linux OS as intermediate layer). The use of Java was agreed for the development of the CEF and the FEF. This led to a consequent adoption of an open source JVM, being IcedTea6, and the lift of usage restriction of the peripheral input/output libraries. Finally the report provided a set of conventional programming styles and software technologies for the design and development of the EF components.

References

[1] Oltmans, E. & A. Lemmen (2006). The e-Depot at the National Library of the Netherlands. The Journal for the Serials Community, 19(1), 61-67.

[2] Altenhöner, R. (2006). Data for the future: 'the German Co-operative development of a long-term digital information archive' (kopal). Library Hi-Tech, 24(4), 574.

[3] Brown, A. (2003). Preserving the digital heritage: building a digital archive for UK Government records. Online Information 2003 Proceedings, 65-68.

[4] Brown, A. (2004). PRONOM 4 User Requirements. The National Archives. Retrieved 20th September 2009, from http://www.nationalarchives.gov.uk/aboutapps/fileformat/pdf/pronom_4_user_reqs.pdf

[5] C64. (2009). "C64" Retrieved 19 September, 2009, from http://www.c64.com/

[6] The Free Software Foundation, GCC, the GNU Compiler Collection. Retrieved 20th September 2009, from http://gcc.gnu.org/.

[7] The Wikipedia Foundations. (2009). "X86" Retrieved 19 September, 2009, from http://en.wikipedia.org/wiki/X86

[8] Lindholm, T. & F. Yellin (1999). Java virtual machine specification, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[9] Microsoft. (2009). Microsoft Windows. Retrieved 19th September 2009, from http://www.microsoft.com/Windows/

[10] Bokhari, S. H. (1995). The Linux operating system. Computer, 28(8), 74-75.

[11] Linux Online Inc. The Linux Home Page at Linux Online. Retrieved 22nd September, 2009, from http://www.linux.org/.

[12] Apple. Mac 101. Retrieved 22nd September, 2009,from http://www.apple.com/support/mac101/

[13] SPARC International. Welcome to SPARC International. Retrieved 22nd September, 2009, from http://www.sparc.org/

[14] IBM. AIX and Unix. Retrieved 22nd September, 2009, from http://www.ibm.com/developerworks/aix/.

[15] Cplusplus.com. (2009). Cplusplus.com - The C++ Resource Network. Retrieved 20th September, 2009, from http://www.cplusplus.com/.

[16] Sun Microsystems, The Java Language Specification, Third Edition. Retrieved 20th September 2009, from http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html.

[17] The Wikipedia Foundations. (2009). Object-Oriented Programming. Retrieved 21th September 2009, from http://en.wikipedia.org/wiki/Object-oriented_programming.

[18] Sun Microsystems. (2009). OpenJDK . Retrieved 19th September 2009, from http://openjdk.java.net/

[19] Kaffe.org. (2009). Kaffe 1.1.9. Retrieved 20th September 2009, from http://www.kaffe.org/doc/kaffe/README.

[20] Gagnon E. M. & Hendren, L. J. (2000). SableVM: A Research Framework for the Efficient Execution of Java Bytecode. Sable Technical Report, McGill University.

[21] Simon, D. and C. Cifuentes (2005). The Squawk virtual machine: Java™ on the bare metal. Conference on Object Oriented Programming Systems Languages and Applications, San Diego, ACM New York, NY, USA.

[22] Sun Microsystems. (2009). Connected Limited Device Configuration (CLDC). Retrieved 19th September 2009, from http://java.sun.com/products/cldc/

[23] Thalinger, C. (2004). Optimizing and Porting the CACAO JVM. Vienna, Vienna University of Technology.

[24] Lougher, R. (2009). JamVM -- A Compact Java Virtual Machine. Retrieved 22nd September 2009, from http://jamvm.sourceforge.net/.

[25] The Free Software Foundation, (2006). GNU Classpath. Retrieved 20th September 2009, from http://www.gnu.org/software/classpath/home.html.

[26] IcedTea6 (2009). Retrieved 22nd September 2009, from http://icedtea.classpath.org/wiki/Main_Page

[27] The Apache Software Foundation. (2009). Apache Harmony - Open Source Java Platform. Retrieved 22nd September 2009, from http://harmony.apache.org/.

[28] Sun Microsystems. (2009). Java 6. Retrieved 19th September 2009, from https://jdk6.dev.java.net/

[29] Sun Microsystems. (2009). Java 7. Retrieved 19th September 2009, from https://jdk7.dev.java.net/

[30] The Free Software Foundation, (2007). GNU General Public License. Retrieved 18th September 2009, from http://www.gnu.org/licenses/gpl-3.0.txt.

[31] Sun Microsystems. (2009). Java SE HotSpot at a Glance. Retrieved 24th September 2009, from http://java.sun.com/javase/technologies/hotspot/

[32] Sun Microsystems. (2009). The Java Compatibility Test Tools. Retrieved 19th September 2009, from http://java.sun.com/developer/technicalArticles/JCPtools/.

[33] XFire. (2009). XFire. Retrieved 19th September 2009, from http://www.xfire.com/

[34] The Apache Software Foundation. (2009). Apache CXF. Retrieved 19th September 2009, from http://cxf.apache.org/

[35] Sun Microsystems. (2009). AWT. Retrieved 19th September 2009, from http://java.sun.com/products/jdk/awt/

[36] Eckstein, R., M. Loy, M., & Wood, D. (1998). Java swing, O'Reilly & Associates, Inc. Sebastopol, CA, USA.

[37] The Free Software Foundation, GCC, the GNU Compiler Collection. Retrieved 20th September 2009, from http://gcc.gnu.org/.

[38] Ritchie, D. M. (1993). The development of the C language. ACM SIGPLAN Notices, 28(3), 201-208.

[39] Rusling, D. A. (1998). The Linux Kernel. Linux Documentation Project. Retrieved on, from url?

[40] The Wikipedia Foundations. (2009). Libraries. Retrieved 19th September 2009, from http://en.wikipedia.org/wiki/Library_(computing)

[41] Birrell, A., D. & Nelson, B. J. (1984). Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, 2, 39-59.

[42] Gudgin, M., M. Hadley Moreau, M., & Nielsen, H. F. (2001). SOAP Version 1.2: W3C Working Draft. Retrieved on 20th September 2009, from http://www.wiscorp.com/SOAPversion1.2W3CworkingDraft.pdf.

[43] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1: W3C Note. Retrieved on 20th September 2009, from http://www. w3c. org/TR/wsdl.

[44] Sun Microsystems. (2009). Code Conventions for the Java™ Programming Language. Retrieved 19th September 2009, from http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

[45] The Eclipse Foundations. (2009). Retrieved 19th September 2009, from http://www.eclipse.org/

[46] Sun Microsystems. (2009). Netbeans. Retrieved 19th September 2009, from http://www.netbeans.org/

[47] The Wikipedia Foundations.(2009). Integrated Development Environment. Retrieved 19th September 2009, from http://en.wikipedia.org/wiki/Integrated_development_environment

[48] The Wikipedia Foundation. (2009). Unified Modeling Language. Retrieved 19th September 2009, from http://en.wikipedia.org/wiki/Unified_Modeling_Language

[49] De Pellegrin, A. (2009). Violet UML Editor. Retrieved 19th September 2009, from http://alexdp.free.fr/violetumleditor/page.php.

[50] UMLet. (2009), Retrieved 19th September 2009, from http://www.umlet.com/

[51] ArgoUML. (2009). Retrieved 19th September 2009, from http://argouml.tigris.org/

[52] The Apache Software Foundation. (2009). The Apache Ant Project. Retrieved 19th September 2009, from http://ant.apache.org/

[53] The Apache Software Foundation. (2009). The Apache Maven Project. Retrieved 19th September 2009, from http://maven.apache.org/

[54] The Free Software Foundation. (2007). GNU Make. Retrieved 18th September 2009, from http://www.gnu.org/software/make/

[55] The Apache Software Foundation. (2009). Apache Log4j. Retrieved 19th September 2009, from http://logging.apache.org/log4j

[56]Sun Microsystems. (2009). Java Logging Overview. Retrieved 19th September 2009, from http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html

[57] Prasetya, W. (2009). JUnit 4.x Quick Tutorial. Retrieved from http://code.google.com/p/t2framework/wiki/JUnitQuickTutorial.

[58] The Wikipedia Foundation. (2009). Unit Testing. Retrieved 19th September, 2009, from http://en.wikipedia.org/wiki/Unit_test

[59] Sun Microsystems. (2009). Javadoc Tool. Retrieved 19th September 2009, from http://java.sun.com/j2se/javadoc/

[60] The Wikipedia Foundations. (2009). Application Programming Interface. Retrieved 19th September 2009, from http://en.wikipedia.org/wiki/Application_programming_interface

[61] W3C. (2009). W3C HTML. Retrieved 19th September, 2009, http://www.w3.org/html/

[62] Pilato, C.M., Collins-Sussman, B., & Fitspatrick, B. (2008). Version control with subversion, O'Reilly & Associates, Inc. Sebastopol, CA, USA.