Project Number	IST-2006-033789
Project Title	Planets
Title of Deliverable	A blueprint for the development of new preservation action tools
Deliverable Number	D2-4.1
Contributing Sub-project and Work-package	PA/2
Deliverable Dissemination Level	External Public
Deliverable Nature	Report
Contractual Delivery Date	30th November 2009
Actual Delivery Date	13th January 2010 (Version 4.1 22nd February 2010)
Author(s)	KB-NL

Abstract

This document contains the blueprint for the development of new preservation action tools. It lists the guidelines that the tools that are designed for or included in the Planets project should offer.

Keyword list: blueprint, preservation action tool, functional requirements, guidelines

Contributors

Person	Role	Partner	Contribution
PA/2-D2, first iteration			
Adrian Brown		TNA	Review
Paul Wheatley		BL	Review
Remco Verdegem		NANETH	Review
Petra Helwig		NANETH	Review
		ALUF	-
Jeffrey van der Hoeven		KB-NL	Review
Maarten Tacoma		KB-NL	Initiation drafts, processing of comments
Caroline van Wijk		KB-NL	Review
Frank Houtman		KB-NL	Review
PA/2-D2, second iteration:			
Caroline van Wijk		KB-NL	Contributor/Review
Frank Houtman		KB-NL	Contributor/Review
Dirk von Sucholetz		ALUF	Contributor
Jefrrey van der Hoeven		KB-NL	Contributor
Remco Verdegem		NANETH	Review
PA/2-D2, third iteration:			
Madelon Hoedt		KB-NL	Author
Wolfgang Keber		MRL	Contributor
Hartwig Thomas		BAR	Contributor
Frank Houtman		KB-NL	Review
Caroline van Wijk		KB-NL	Review
PA/2-D2, this iteration:			
Sara van Bussel		KB-NL	Author
Lynne Montague		TNA	Contributor/Review
Andrew Jackson		BL	Contributor
Bill Roberts		NANETH	Review
Michael Carden		National Archives of Australia	Review

Document Approval

Person	Role	Partner
PA/2-D2, first iteration:		
Hilde van Wijngaarden	Internal reviewer	KB-NL
PA/2-D2, second iteration:		
Barbara Sierman	Internal reviewer	KB-NL
PA/2-D2, third iteration:		
Frank Houtman	Internal reviewer	KB-NL
PA/2-D2, this iteration:		
Bill Roberts	External reviewer	NANETH

EXECUTIVE SUMMARY

Software programs, developed to perform preservation actions, will become part of the Planets project. As the development of these tools is spread among different companies or institutions, different techniques and standards might be used. To ensure a consistent behaviour of the Planets system as a whole and a consistent level of quality, there are a number of guidelines to functionality that all preservation action tools used in the project should offer. This document lists these guidelines for migration and emulation tools that are to be developed. The blueprint can be used by developers outside of Planets.

This final iteration contains a final list of guidelines. This document is based both on comments from other Planets partners as well as two case studies (of an emulation tool and a migration tool), which are not included.

Alterations since last iteration

In the table below, changes between the previous iteration of the Blueprint deliverable and the current iteration are listed.

Modification	Description	Date
Updating of information	Updating information to the current developments within Planets.	7-12-2009
Filling in gaps and incorporating comments	Resolving all comments and gaps from previous iterations.	7-12-2009
Comparison of Blueprint to Planets Core Registry		7-12-2009

TABLE OF CONTENTS

1. Document overview

1.1 Purpose of this Document

The purpose of the blueprint is to provide guidelines for new preservation action tools developed outside of the Planets project, which might be created specifically for, or will be integrated into the Planets framework. Preservation action tools can be tools for objects and tools for environments.

1.2 Scope of this Document

This document consists of guidelines for the development new preservation action tools and is not a true blueprint in that sense. A tool that does not confirm to every guideline will still be considered for inclusion within the Planets Framework. If a tool does comply with all guidelines this does not automatically imply Planets approval of the tool, the tool will still need to be tested. Therefore these guidelines need to be regarded as guidance for tool development or optimalization.

Detailed information about tool inclusion into the Planets framework and the wrapping process is not covered in this document and will be published on the Planets website before the end of the project.[1]

1.3 Context

Within the Planets project, a number of software programs will be used to perform preservation actions. These tools can be manufactured by different software developers and might make use of a variety of techniques and standards in order to accomplish their task. However, to ensure a consistent user experience and a good overall quality of operation there are a number of guidelines that preservation action tools used within the Planets project should comply with. The blueprint aims to be a working document to provide a list of these guidelines.

As mentioned in the scope, this document focuses on the guidelines for preservation action tools, as technical and other Planets-specific requirements might be provided by the other subprojects within the Planets project.

1.4 Document overview

This document is divided into four chapters. The first chapter gives a general outline of the contents of the document. In chapter two, it is followed by an explanation of the methodology used in the creation of this document. In chapter 3, the guidelines for preservation action tools that have been identified are listed.

[1] Planets website: http://www.planets-project.eu

2. Methodology

2.1 Introduction

This chapter provides an overview of the methodology used so far in defining the guidelines for tools.

2.2 Design Overview

Many companies have a set of requirements that the software they produce has to comply with. Within the Planets project, the development of preservation action tools is not limited to one company but spread among development teams worldwide. To ensure a consistent behaviour of the Planets system as a whole, the preservation action tools developed for and integrated in the project should to comply with a set of guidelines.

Traditionally, a distinction is made between the functional and the non-functional requirements of a system. Non-functional requirements are criteria that can be used to judge the operation of a system, but do not describe any specific feature, while functional requirements specify specific behaviours of the system. To ensure a broad coverage of features, the list of non-functional requirements that's presented in ISO-9126[1] has been taken as a starting point in compiling the list of guidelines that's presented in chapter 3 of this document.

These non-functional requirements are very general in nature, and not all are equally relevant for describing preservation action tools. When a non-functional requirement was thought to not be relevant, it has been ignored.

To make sure the currently defined list guidelines is sufficient for accurately describing the desired functionalities of preservation action tools an empirical method has been used. The experience gathered in incorporating preservation action tools in the Interoperability Framework and Testbed, as well as the outcomes of the analysis of currently available preservation action tools by the PA/4 and PA/5 working groups has been used to validate and extend the current set of guidelines.

The guidelines have also been compared to the Planets Core Registry, to the Software Packages entity and its underlying entities. From this comparison no new guidelines were discovered.

3. Guidelines for PA tools

3.1 Introduction

In this chapter, the guidelines for preservation action tools – to be developed outside and within the Planets project - are presented. The guidelines are divided into groups within the non-functional requirements defined by the ISO-9126-standard.

The guidelines for tools for objects and tools for environments have not been merged. Testing with the use of case studies showed that such a merger was not feasible for all guidelines because of the differences in the functionality of these different types of tools. Different sections for both migration and emulation are present for functionality. The other tables (reliability, usability, etc.) do not have such a division and apply to both migration and emulation tools. The tables found in this chapter apply to tools for objects (3.2.1) and tools for environments (3.2.2), respectively.

When talking about tools for environments, these guidelines are written for emulators that emulate a hardware environment. Any emulators that work in a different manner (such as the Universal Virtual Computer) might find that not all guidelines apply to them.

3.2 Guidelines tables

In the tables below the guidelines for preservation action tools – tools for objects and tools for environments – to be used within the Planets project are listed. In the second column, the non-functional requirements as defined by ISO-9126 are used for categorizing the guidelines in the fourth column. Although some of the non-functional guidelines appear to be less applicable to preservation action tools than others, this methodology has been used to ensure a broad coverage.

Guidelines such as ‘Resource behaviour’ (‘The tool should not consume more than X% of processor power when running in the Testbed environment.’) have been removed from the list of guidelines because this turned out to be not a requirement or limitation for the Testbed or the Interoperability Framework.

To indicate the type of guideline, a number of keywords are used as defined by the IEEE in the Standards Style Manual [2]. In the table below, the words that command the presence of some feature or function are listed in decreasing order of strength.

Shall	used to indicate preferred guidelines
Should	used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others
May	used to indicate a course of action permissible within the limits of the standard
Can	used for statements of possibility and capability

3.2.1 Guidelines for tools for objects

Table 1.1: Guidelines

ID	Category	Sub-Category	Guideline	Comments
	Functionality			
M1.1		**Suitability**	The tool **shall** be able to read, convert and write one or multiple file formats.	
M1.2			The tool **should** follow the guidelines given in the official file format specification when dealing with a certain file format.	
M1.3			When reading or writing a certain file format, whenever the tool's behaviour differs from what the file format specification for that format prescribes, this behaviour **shall** be clearly described in the user manual.	
M2.1		**Accuracy**	If a tool reads a file and finds chunks of data that it cannot interpret (but that does not prevent it from interpreting the rest of the file), it **shall** report this as a warning in the log file.	
M2.2			If the tool can only partly interpret a file format (i.e. it can read the format but not all properties are supported), any occurrence of a property that couldn't be interpreted **shall** be written to the log file since important data could be lost.	
M3.1		**Interoperability**	The tool **shall** be able to operate within a Service Oriented Architecture, or have a clear interface.	

3.2.2 Guidelines for tools for environments

Table 1.2: guidelines

ID	Category	Sub-Category	Guideline	Comments
	Functionality			
E1.1		**Suitability**	The tool **shall** be able to read, convert and write one or multiple file formats.	
E1.2			The tool **shall** follow the original specifications of the hardware components. Or, if not available, **shall** make a best interpretation of the original behaviour of the hardware component.	
E2.1		**Accuracy**	The tool **shall** be able to emulate the target hardware environment as accurately as possible.	
E2.2			Specific known limitations of the tool concerning the emulation of the original hardware environment **shall** be mentioned in the tool description.	
E3.1		**Interoperability**	The tool **shall** be able to operate within a Service Oriented Architecture, or have a clear interface.	
E4.1		**Compliance**	The emulated hardware environment **should** comply with the standards, guidelines and protocols that are available for the original hardware.	IBM PC, Intel Architecture, ATA, etc.

3.2.3 General guidelines for preservation action tools

Reliability		
G1.1	**Maturity**	If the user attempts to open a file in a file format the program is not supposed to be able to open, the program **shall** react with an error message.
G1.2		If the user attempts to open a file that cannot be read because the file size is too large, the program **shall** react with an appropriate error message.
G1.3		If the user attempts to open a file that cannot be read because the file size is too small to contain a valid file of a certain format, an appropriate error message **shall** be given by the program.
G1.4		If the user attempts to open a file that is valid according to the specification, but doesn't contain any data, the program **shall** give an appropriate error message.
G2.1	**Fault tolerance**	The tool **shall** provide a human-readable appropriate error message in the English language in case of an error.
G2.2		In case of an error, an error code **shall** be provided that identifies the error and makes it possible for other programs to react on the situation.
G3.1	**Recoverability**	Errors **shoul**d be logged, together with a unique identifier.
G3.2		At least an administrator **shall** be able to consult the log file.
G3.3		The tool **shall** keep a log of all important operations it performs. This log should include the time the event took place and a human-readable description of the event itself.
Usability		
G4.1	**Understandability**	A human readable description of the tool in the English language **should** be included in the service registry.
G4.2		The status of the tool **should** be made clear to the user via messages and progression bars.
G4.3		The tool **shall** offer a user-friendly and intuitive

		interface.	
G4.4		The user interface **should** either be graphical (GUI) or command line.	For an emulator, two user interfaces will be in place: the first, the emulated interface, and secondly, the user interface of the tool itself. The latter **should** be a GUI.
G4.5		The user interface **may** support multiple languages.	
G5.1	**Learnability**	An installation guide **should** be provided. This should at least be available in the English language.	
G5.2		A user manual **should** at least be available in the English language.	
G5.3		The user manual **may** contain a description of the most common error messages that may appear during usage of the tool.	
G5.4		Context dependent online help functionality **should** be available.	
G6.1	**Operability**	It is recommended that the tool can be invoked as a web service. Furthermore, a client-side installation of the tool **should** be possible.	
G6.3		The tool **shall** offer a machine-readable interface.	The interface can be used for exchange of information packages, emulator commands, and configuration actions.
Efficiency			
G8.1	**Time behaviour**	The tool **shall** behave as naturally as possible. Timing mechanism may be delayed but should stay synchronised.	
Maintainability			
G10.1	**Analysability**	The source code **shall** be well formed, documented and understandable.	
G10.2		The tool **may** offer a debugging facility to closely analyse the flow of execution and memory usage.	

G10.3		The source code **may** be made public or can have restrictions for reuse and distribution.	Although Planets can not force creators to make their code open source, it recommends doing so.
G11.1	**Changeability**	The tool **shall** conform to a well defined versioning procedure.	
G11.2		The tool **may** offer several versions via a controlled versioning environment.	
G11.3		The tool **should** be adaptable to correct errors and adjust the tool to a changing environment.	
G12.1	**Stability**	The risk of unexpected effect of modifications **should** be reduced as much as possible.	
G12.2		The tool **shall** not harm the underlying (host) hardware environment.	Although malfunctions may not be caused by the tool itself (for example, Windows may crash), the tool should not have a negative influence on the rest of the system.
G14.1	**Compliance**	Versioning procedures **shall** be followed.	
Portability			
G15.1	**Installability**	The tool **should** not be dependent on third-party decompression software in order to be installed.	
G15.2		The tool **shall** be easy to install and configure on the host hardware environments defined in the installation manual.	
G15.3		The target hardware environment **should** be easy to setup via machine readable configuration and/or human readable configuration.	This may depend on many other tools/services.
G15.4		The target hardware and software **can** be setup and running automatically.	
G15.5		Required target software (OS, applications etc.) **shall** be independent of physical data carriers.	
G16.1	**Co-existence**	Multiple instances of the tool **may** be installed and run simultaneously.	
G17.1	**Replaceability**	Uninstalling or upgrading the tool **shall** be easy.	

Appendix A : References

[1] ISO/IEC 9126-1:2001 Software engineering -- Product quality -- Part 1: Quality model

[2] IEEE 2000 -- IEEE Standards Style manual