SCAPE
Scalable Preservation Environments

Final version of automated policy-aware planning component

Authors
Michael Kraxner, Markus Plangg (Vienna University of Technology)

July 2014

This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).
This work is licensed under a CC-BY-SA International License

Executive Summary

This document accompanies the prototype implementation of the automated planning component Plato. It describes how Plato integrates with and enhances the SCAPE ecosystem, increasing efficiency and trustworthiness of preservation planning.

Table of Contents

1 Introduction

Digital content holders have to ensure their assets are preserved and kept accessible for their communities in the long term. This requires continuous monitoring of the environment (results of operations, formats in use, or new emerging formats and tools), planning, and execution of corrective actions necessary to minimize risks and ensure continuous access.
These processes need to be integrated with each other and with digital repositories. The set of integrated digital preservation processes comprise the preservation lifecycle.

A range of tools to support these processes have emerged in recent years. The PLANETS project defined a trustworthy preservation planning process [1] implemented in the preservation planning tool Plato 3 [2]. Plato guides the user through the planning workflow, collecting necessary information, helping her to arrive at an informed decision, producing a well-documented plan.
Planning with Plato 3 was a labour intensive endeavour requiring substantial human effort:

1. Information describing the environment such as institutional policies had to be entered manually, and was only retained as documentation.
2. Selection of representative samples was left to a user to define manually.
3. Decision criteria were often created ad-hoc hindering knowledge extraction and reuse.
4. Registries for migration tools were integrated, but due to their closed nature they were hard to maintain beyond the end of the project.
5. Prototype implementations for automatic evaluation of migration results with respect to the decision criteria were present. However, only as part of Plato and therefore they were not available to ensure quality of results during operations.
6. Generation of an executable plan was done as very simple proof of concept for one repository only.
7. Plato 3 lacked collaborative features: Planning was performed on a one-user-per plan basis. The only way to share a plan was to make it public, beside that it could be exported, to pass on a snapshot of its current state.

The automated planning component builds on the planning tool Plato 3. The main goals are to:

1. Improve efficiency based on existing knowledge.
2. Align planning with institutional policies.
3. Create actionable preservation plans.
4. Provide mechanisms to create and maintain continuously evolving preservation plans.

The document is structured as follows. Section 2 describes the context of the planning component, its role in the preservation lifecycle, and the importance of a common language and formalized policies. Section 3 presents a taxonomy for decision factors and an analysis tool. Section 4 discusses the improvements to the preservation workflow.
Note that this document contains content sourced from recent publications.

1.1 Preservation Lifecycle

The SCAPE Planning and Watch suite [3] was developed to provide an open, scalable environment for preservation control and monitoring. It builds on the conceptual foundation of Plato, supports step-wise integration of systems through open interfaces, and enables organisations to practically apply Planning and Watch in a scalable, semi-automated way.

Figure 1 illustrates how the components of suite interact to support the preservation lifecycle.

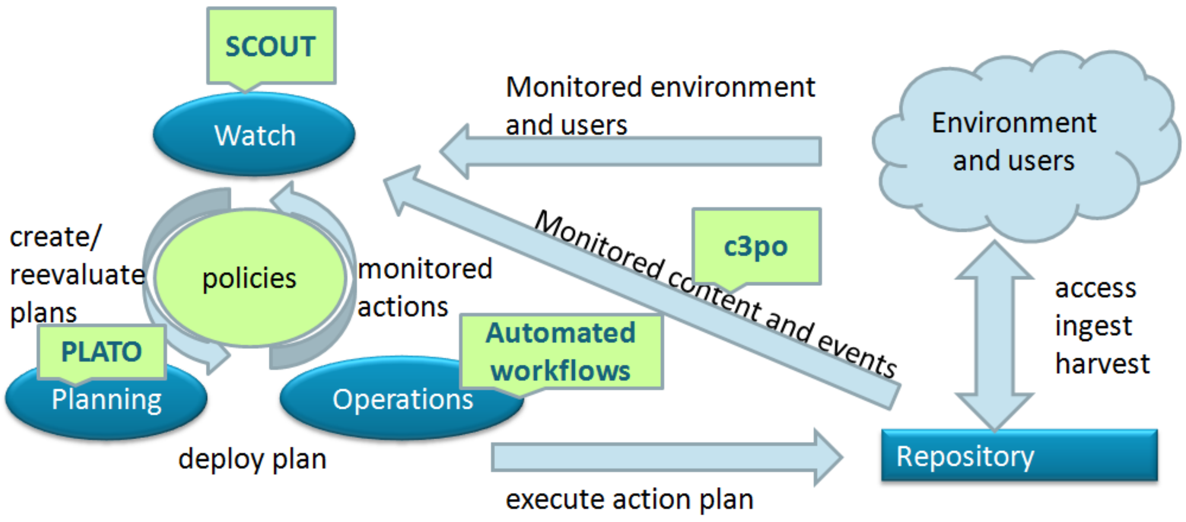


Figure 1 Preservation lifecycle [3]

The lifecycle starts with a repository containing content preserved for a designated community over time. Beyond this community there are a variety of factors of potential interest to be monitored, including other repositories, technical solutions, and format risks as well as other aspects.
In order to analyse a collection, the repository characterizes the content passing the metadata to the scalable content profiling tool C3PO[1], which creates a content profile that can be exported as XML.

This content profile is linked by the Automated Watch component, Scout[2], to its internal data model and can be matched against an organization's control policies. Scout can detect policy violations such as format profiles that pose specific risks, the presence of risk factors such as compression, or other conditions that are of interest and require mitigation.

Upon discovering a condition that requires intervention, Scout notifies the responsible decision maker, who analyses the situation guided by Plato. The result is a preservation plan, which contains an executable plan. It can be directly deployed to the configured endpoint of the repository, where it can be run on the original content set from which the content profile was derived. These operations are monitored for compliance to the service level agreements, which is performed by the monitoring component Scout, completing the first round of the preservation lifecycle.

1.2 Analysis and Taxonomy for Decision Factors

A public instance of Plato 3 was made available during the PLANETS project. It has been used for educational purposes, and by institutions carrying out real-world case studies and pilot-test cases

[1] https://github.com/openplanets/c3po
[2] https://github.com/openplanets/scout

resulting in a growing body of knowledge regarding preservation decisions. Unfortunately reasoning about this accumulated knowledge and sharing of experience was hindered by a lack of formality in the representation of decision criteria.

This issue was addressed in the SCAPE deliverable D14.1 [5]: It presents a taxonomy for decision factors that provided input for the quality vocabulary and enables systematic analysis of decision factors.
The Knowledge Browser was developed providing a graphical interface used to analyse decision factors of plans, their interdependencies and co-occurrences, and evaluate their importance for specific preservation cases. It is part of the Planning Suite and is deployed along with Plato 4.

1.3 Quality Vocabulary and Control Policies

In *Open Preservation Data: Controlled vocabularies and ontologies for preservation ecosystems* [6] the authors point out the need for a common language to connect the components of a scalable preservation system. They present a semantic model for organisational policies and objectives that represents the drivers and constraints for preservation processes using an extensible ontology.
A vocabulary to express properties of quality[3] is defined including a catalogue of measures[4], which is based on the analysis of decision factors discussed in the previous section.
These measures are used to facilitate publishing, discovering, and automatic composition of preservation components.
Besides, they are the basis to express organizational objectives in a machine understandable way as *control policies*, which comprise:

- Content sets: Represent a collection of objects that are the focus of the policy.
- User communities: The community for whom the digital content is preserved for.
- Preservation cases, which relate content sets and user communities with objectives.
- Objective: a simple statement about a property of e.g. a format, tool, or representation of content. It relates to a measure of the vocabulary and specifies its desired or inadmissible value range.

Control policies are also described in more detail in deliverable D13.1 [4].

1.4 SCAPE Preservation Components

The SCAPE project uses Taverna as its workflow system. It allows manual creation of workflows and composition of existing ones via the graphical interface of the Taverna Workbench. Once completed, they can be executed on a Taverna Server. Workflows and workflow fragments can be shared with other people from the community via the social workflow sharing platform myExperiment[5].
For the SCAPE project myExperiment has been extended with the concept of Component Profiles [7]. These specify workflows with standardised inputs and outputs, which are semantically annotated to ease discovery and enable automatic composition via unified interfaces.

3 http://purl.org/DP/quality
4 http://purl.org/DP/quality/measures
5 www.myexperiment.org

There are four component profiles for preservation components: Characterisation, quality assurance, migration, and executable plan[6].
Migration components take the source object as input and provide the migrated object as output. The characterisation and quality assurance profiles prescribe the objects as input. The output consists of characterisation and quality assurance measures from the measure catalogue.
These components provide building blocks for more complex preservation actions. When combined to an executable plan as shown in Figure 2, this resulting component can be executed on the SCAPE execution platform.

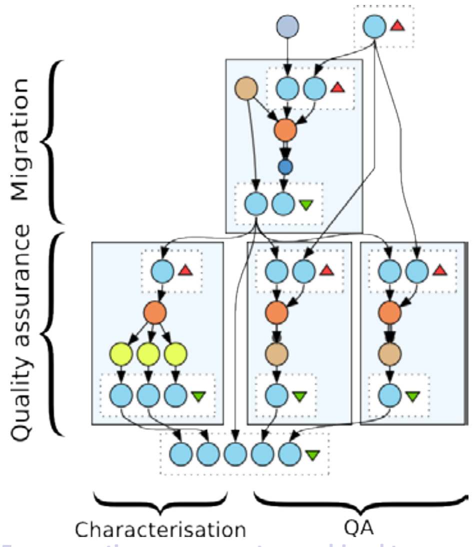


Figure 2 SCAPE preservation components, combined to executable plan [7]

2 Efficient and Trustworthy Preservation Planning

In 2013 a case study was conducted together with the Danish State and University Library to evaluate the efficiency of the preservation planning process with Plato 3 [8]. It showed that specifying the requirements, choosing meaningful samples, and conducting a structured evaluation and documenting results are the most time consuming steps and that there is great potential to increase efficiency.

Preservation planning is dependent on information from different sources and comprises a range of tasks. Due to its wide scope it is neither feasible nor advisable to cover all functionality directly. Instead Plato 4 aims to extend the existing planning tool by integration of existing tools through open interfaces, and facilitate knowledge sharing and reuse through adoption of a common language.

Plato connects to any SCAPE repository – that is any repository implementing the SCAPE APIs - via Data Connector API and Plan Management API, understands control policy definitions and extensively uses the vocabulary described above: As soon as an organization has defined its objectives as control policies they can be utilized for the planning process.

The following sections explain how these measures enhance the preservation planning workflow.

[6] http://openplanets.github.io/scape-component-profiles

2.1 Define Basis

The first step is to describe the environment in which the preservation process takes place. This includes information about the organization, its policies, the collection and the reasons for preserving it, and the designated community.

If control policies have been defined for the planner's user group in Plato, one of the preservation cases can be selected. Related information about the environment is automatically assigned to the corresponding fields in the plan. The control policies related to the selected preservation case are also documented in textual form, and furthermore enable automation in the later steps.

2.2 Define Sample Objects

In this step metadata about the collection to be preserved is gathered and analysed and representative samples are selected as a basis for evaluating preservation actions.
This requires a deep understanding of the collection to be preserved. Manual analysis is a challenging task even for small collections, as a wide range of characteristics ha to be considered. As an example seemingly similar files regarding their file type can bear different problems for migration tools when they have a certain size, or specific features like number of pages or data tables in case of electronic documents [9].

C3PO enables in-depth analysis of a repository's content through aggregation and analysis of characterization data. A web-front end visualises the data and allows:

- Interactively filtering and examining the collection,
- Derivation of representative sample sets, using different sample selection algorithms, and
- Exporting the result as content profile.

Content profiles from C3PO are integrated directly with Plato 4. Therefore both, the technical processes of characterization and sample selection, as well as analysis of content sets are automated. A profile derived from a collection held by a SCAPE repository can be processed by Plato 4 which can retrieve samples via the Data Connector API and store them in the plan for later experiments. This also requires that the user has the appropriate permissions and credentials and configures her group so Plato can connect to the repository.

Independent of an existing connection to a Data Connector endpoint the aggregated information about the collection is applied to the plan, this comprises:

- The number of objects,
- A summary on the type distribution,
- The algorithm used to determine representative samples, and
- Definition of sample objects: For each representative sample one sample object is added with name and characterization information from the collection profile. (Without connection to a repository the sample itself cannot be added to the plan, thus automatic experiments will not be available. Nevertheless the planner does not need to enter this information manually.)

2.3 Identify requirements

The main goal of preservation planning is to determine which alternative best satisfies the preservation requirements of a particular collection. An objective comparison requires collecting all requirements and breaking them down to measurable criteria. This is a demanding task needing a detailed understanding of the institutional environment, e.g. policies, legal obligations, financial constraints as well as specialist technical knowledge concerning file formats and tools.

Control policies already define constraints, e.g. acceptable formats, tool performance; modelled as objectives.
Plato 4 is policy aware and uses these control policies to reduce the effort required as far as practical. If a preservation case was selected in step Define Basis, the objective tree is built automatically by deriving decision criteria from all contained objectives. The scale of each criterion is derived from the measure, and a mapping from criterion to measure is established. The modality of the objective can be used to prepare utility-functions to assist in step *Analyse Results.*
Plato 4 also allows the user to manually map decision criteria to measures from the vocabulary.

When mapped to measures, either manually or automatically, decision criteria enable automated quality assurance and evaluation of migration results, described later and allow reasoning about different plans using the Knowledge Browser presented earlier.

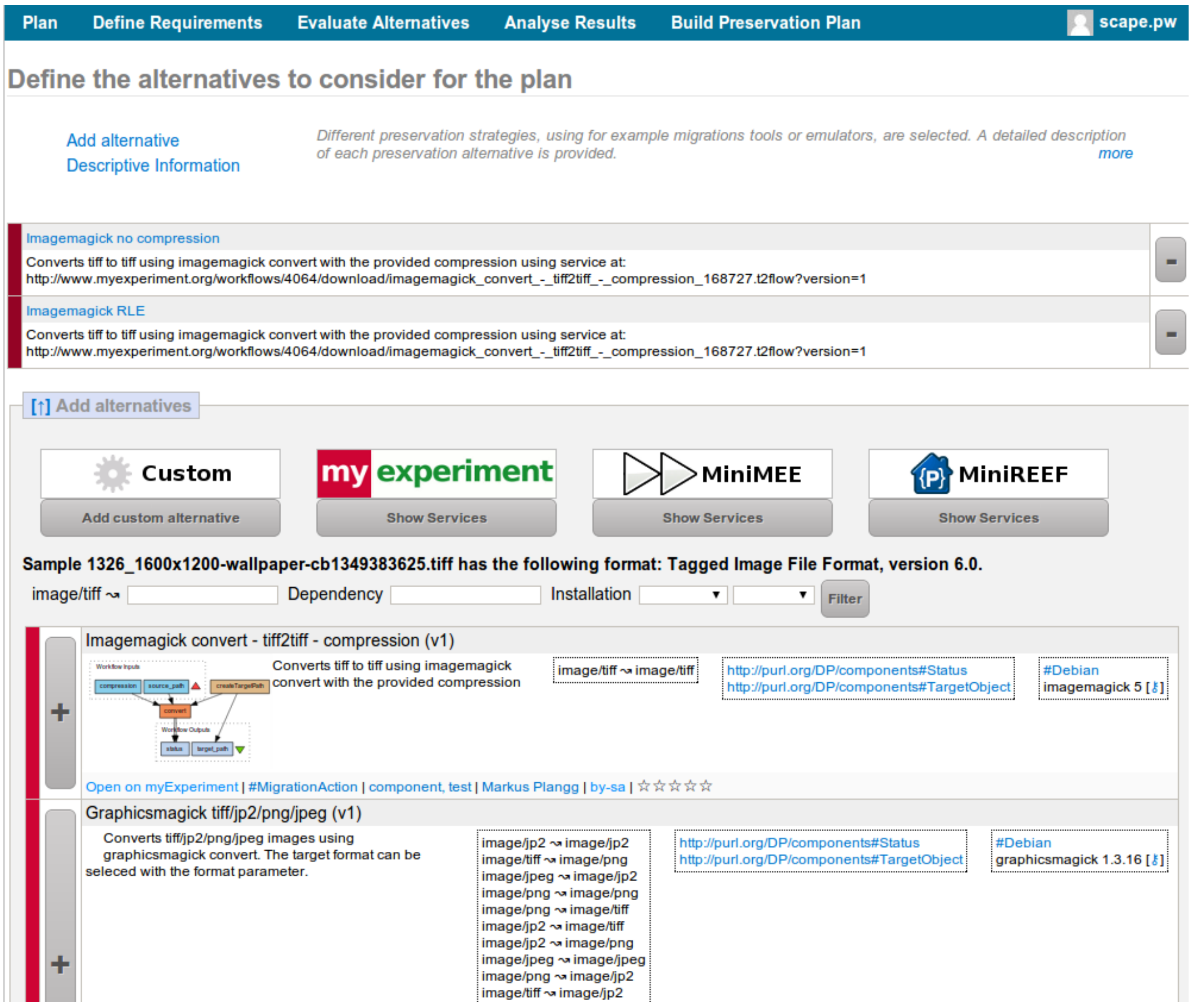
2.4 Define Alternatives

Rather than developing new registries for preservation actions, Plato 4 strives to employ existing solutions and communities to facilitate experimentation. Plato integrates with the workflow sharing platform myExperiment where workflows can be published and discovered to enable reuse by others in the community.

Workflows foster reproducibility, a key requirement for trustworthy and evidence based preservation planning. They provide detailed descriptions about the involved tools and their dependencies, the expected input data, and produced results. Information about the dependencies of these tools is also important for later operational deployment.

Figure 3 shows how Plato 4 integrates with myExperiment. Plato queries for components that can migrate the previously selected sample objects based on their MIME type. Further filters can be applied, like restricting components for certain target formats, specific target platforms or distributions thereof.

To support the selection of actions additional information is presented: Regarding its credibility the creator and a reputation score, and the applying license.



Define the alternatives to consider for the plan

Add alternative
Descriptive Information

Different preservation strategies, using for example migrations tools or emulators, are selected. A detailed description of each preservation alternative is provided. more

Imagemagick no compression
Converts tiff to tiff using imagemagick convert with the provided compression using service at:
http://www.myexperiment.org/workflows/4064/download/imagemagick_convert_-_tiff2tiff_-_compression_168727.t2flow?version=1

Imagemagick RLE
Converts tiff to tiff using imagemagick convert with the provided compression using service at:
http://www.myexperiment.org/workflows/4064/download/imagemagick_convert_-_tiff2tiff_-_compression_168727.t2flow?version=1

[↑] Add alternatives

Sample 1326_1600x1200-wallpaper-cb1349383625.tiff has the following format: Tagged Image File Format, version 6.0.

image/tiff ↝ Dependency Installation Filter

Imagemagick convert - tiff2tiff - compression (v1)

Converts tiff to tiff using imagemagick convert with the provided compression

image/tiff ↝ image/tiff

http://purl.org/DP/components#Status
http://purl.org/DP/components#TargetObject

#Debian
imagemagick 5 [⚷]

Open on myExperiment | #MigrationAction | component, test | Markus Plangg | by-sa | ☆☆☆☆☆

Graphicsmagick tiff/jp2/png/jpeg (v1)

Converts tiff/jp2/png/jpeg images using graphicsmagick convert. The target format can be seleced with the format parameter.

image/jp2 ↝ image/jp2
image/tiff ↝ image/png
image/jpeg ↝ image/jp2
image/png ↝ image/png
image/png ↝ image/tiff
image/jp2 ↝ image/tiff
image/jp2 ↝ image/png
image/jpeg ↝ image/jpeg
image/png ↝ image/jp2
image/tiff ↝ image/jp2

http://purl.org/DP/components#Status
http://purl.org/DP/components#TargetObject

#Debian
graphicsmagick 1.3.16 [⚷]

Figure 3 Querying components from myExperiment

Note that alternatives can still be defined manually, or added from other sources, but they lack:

- formal documentation of the process,
- support for automatically execution, and
- cannot be used to generate executable plans automatically.

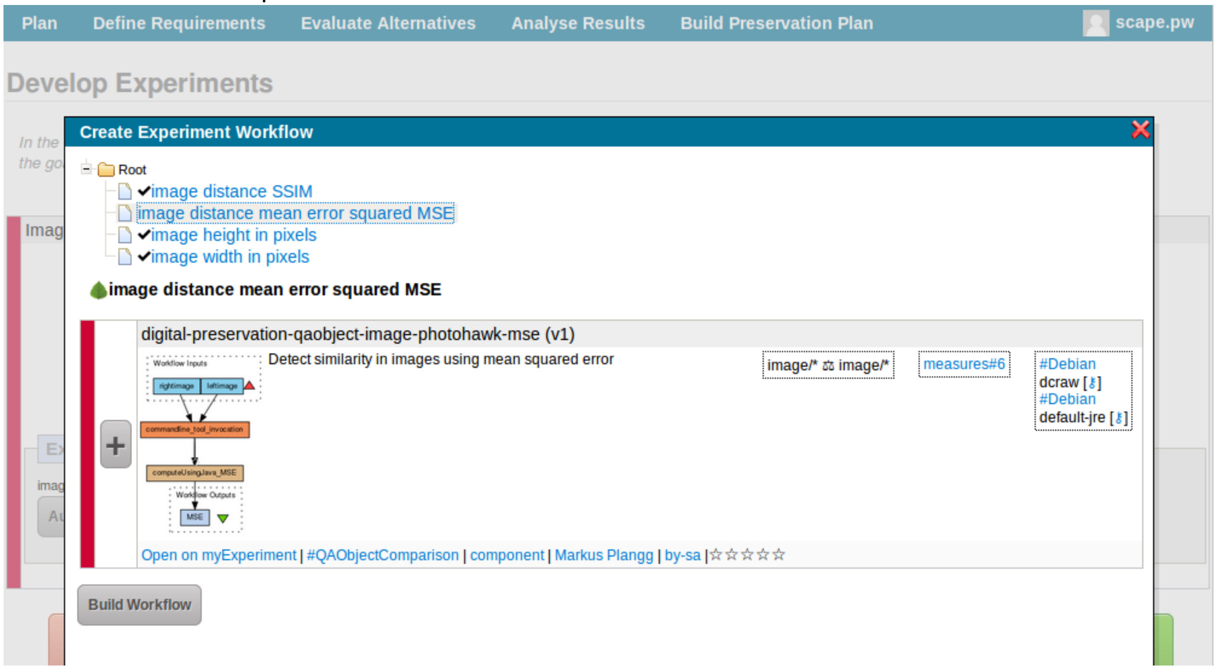
2.5 Develop Experiments

For alternatives based on migration components, Plato 4 supports automated experiment generation. The resulting Taverna workflow is a fully annotated executable plan component incorporating migration, characterisation and quality assurance.

Starting from the decision criteria defined in Section 2.3, characterization and quality assurance components are looked up on myExperiment. Characterisation components are connected to the output of the migration component to extract characteristics of the migrated object. Quality assurance components are connected to the source object and the migrated object. Relevant output ports are created and connected to the components.

To allow more fine grained control over the experiment creation, components can also be selected manually. For each decision criteria, relevant components are queried and presented to the user (see Figure 4).

The resulting workflow adheres to the component profile Executable Plan and can be used without modifications for the production environment.


Plan | Define Requirements | Evaluate Alternatives | Analyse Results | Build Preservation Plan | scape.pw

Develop Experiments

Create Experiment Workflow

Root
- ✓image distance SSIM
- image distance mean error squared MSE
- ✓image height in pixels
- ✓image width in pixels

image distance mean error squared MSE

digital-preservation-qaobject-image-photohawk-mse (v1)

Detect similarity in images using mean squared error

image/* ⚖ image/* | measures#6 | #Debian dcraw [⚷] #Debian default-jre [⚷]

Open on myExperiment | #QAObjectComparison | component | Markus Plangg | by-sa | ☆☆☆☆☆

Build Workflow

Figure 4 Fine grained component selection for experiments

2.6 Build Preservation Plan

Controlled experimentation and objective evaluation of the Components leads to selection of the most suitable preservation action. To ensure trustworthiness the component used in the production environment should be identical with the one under test.

Additional information is needed to deploy and run the plan on the Execution Platform. This information is aggregated in the Preservation Action Plan, an XML document conforming to a lightweight XML-Schema[7] to allow interoperability with the Execution Platform. It contains information regarding:

- **Objects**: It identifies the objects that the preservation action should be applied to. These might be identified by enumeration of object IDs, or a Search/Retrieval via URL (SRU) query that selects a subset of the collection. This information is used to request the content from a Data Connector API implementation and is taken from the content profile.
- **Executable Plan**: When working with Components, Plato already generates an Executable Plan for evaluation - as described in Section 2.5.
- **Quality Level Descriptions**: Constraints on the quality of the output used to decide if the result is acceptable. Section 2.7.1 describes how these can be derived from the decision criteria.

The Preservation Action Plan is stored as part of the preservation plan for documentation purposes. Plato itself does not trigger plan execution, but is integrated with the SCAPE Plan Management API [10]. Once a plan is approved by the planner, it can be deployed to the repository using a Plan Management endpoint, and enabled via the Plan Management GUI.

[7] http://ifs.tuwien.ac.at/dp/plato/schemas/preservationActionPlan-V1.xsd

2.7 Monitoring of results

During the planning process, a set of decision criteria are defined. These evaluate the important aspects of the data to be preserved, the environment, and the actions to be applied.
Based on these criteria, alternative preservation actions are evaluated and a ranking is calculated. The planner can then choose the best suited action and adopt it.

Depending on the measure, the measurements can result in values of different scales. Before these values can be aggregated to calculate the score of an action, they have to be transformed to a uniform target scale. Plato uses target scale [0, 1...5] where 0 has a special meaning. With multiplicative aggregation one value of 0 will set the overall score of the action to 0
Therefore:

- If the target scale of a criterion includes 0, this means that this measure MUST (/NOT) have a certain value.
- Likewise a target scale from 1-5 could be read as a suggestion: this measure should have a certain value (the closer to 5 the better)

The aggregated scores lead to the adoption of a certain action; therefore the related measures should be monitored during operations to track whether the action continues to perform according to expectations.

2.7.1 Quality Level Definitions

Quality Level Definitions (QLD) describe the expected properties of migrated representations. They relate to criteria such as whether the created files are well-formed. For criteria which MUST (/NOT) have a certain value during planning, a violation means that the migrated object is not valid and should not be fed back to the repository. Thus validation should happen directly after migration to avoid unnecessary transfer of objects.
Based on the measures and the defined transformers Plato generates QLDs encoded as Schematron[8] schemas. These are added to the Preservation Action Plan and can be used by the Execution Platform to validate quality assurance and characterisation outputs. (See Section 6.1 for an example.)

2.7.2 Integration with Scout

Decision criteria which relate to the expected behaviour of components (e.g. their runtime) or to non-dynamic aspects, e.g. related to the format such as the ISO standardisation of PDF versions, can be monitored as risks and opportunities using Scout. It uses an RDF triple store to manage the aggregated knowledge. Questions – so called *triggers* - can be defined using SPARQL queries and are periodically executed in defined time intervals. When the query returns a result, a notification is sent to the planner.
Plato generates triggers based on information in decision criteria. Mapped measures specify the measurements to look for, and defined transformations can be used to derive acceptable value ranges. The triggers can be stored directly in Scout after the plan has been deployed to the repository.

[8] http://www.schematron.com/

2.8 User management

In previous Plato versions account sharing was the only way to work collaboratively on a preservation plan. This does not reflect complexity inherent to the preservation planning process, where workflow steps require knowledge and expertise of different fields.
Therefore group management was added to Plato 4. It enables users to define settings like the repository which should be used, the endpoint of the watch component, as well as the control policies of the organization on a per group basis.

For security and identity management the Identity Provider (IDP) was added to the planning suite. It allows single sign on between the applications of the Planning Suite, and basic features like account creation and password reset in a safe way. It is based on PicketLink[9] and allows integration with existing identity management systems of organizations.

3 Maintenance and Sustainability

Following the SCAPE open source development guidelines Plato is released under the terms of the Apache License Version 2.0. A major goal was to increase accessibility of the project to other developers, facilitate both the development process and setting up of the environment.
Apache Maven is used for project configuration and dependency management, and eases configuration of automated tests and generation of quality metrics.
Git is used as version control tool for the source code, and GitHub[10] as hosting platform, to organize the development process, and provide information how to participate.

The Planning Suite comprises of a set of web applications, the Identity Provider, Knowledge Browser, and Plato, and is deployed as Enterprise Archive on a JBoss AS 7[11].

The latest version of the Planning Suite is hosted as a public instance. It can be reached from the main Plato web site[12], which also provides thorough documentation and background information.
In addition to this main instance, which will be maintained beyond SCAPE by TUW, a configuration for Vagrant[13] is provided that allows to quickly set-up a development environment and supplements the textual guide how to set up the system.

4 References

1. *Systematic planning for digital preservation: Evaluating potential strategies and building preservation plans.* **Christoph Becker, Hannes Kulovits, Mark Guttenbrunner, Stephan Strodl, Andreas Rauber, Hans Hofman.** s.l. : International Journal on Digital Libraries (IJDL), 2009.
2. *Plato: a service-oriented decision support system for preservation planning.* **Christoph Becker, Hannes Kulovits, Andreas Rauber, Hans Hofman.** s.l. : Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'08), 2008.

[9] http://picketlink.org/
[10] https://github.com/openplanets/plato
[11] http://jbossas.jboss.org/
[12] http://ifs.tuwien.ac.at/dp/plato/
[13] http://www.vagrantup.com/

3. *The SCAPE Planning and Watch suite.* **Michael Kraxner, Markus Plangg, Kresimir Duretec, Christoph Becker, Luis Faria.** Lisboa, Portugal : 10th International Conference on Preservation of Digital Objects (IPRES 2013), 2013.
4. **Bechhofer, Sean.** *Final version of Policy specification model.* s.l. : SCAPE Project Deliverable (D13.1), 2013.
5. **Markus Hamm, Christoph Becker.** *Report on decision factors and their influence on planning.* s.l. : SCAPE Project Deliverable (D14.1), 2011.
6. *Open Preservation Data: Controlled vocabularies and ontologies for preservation ecosystems.* **Hannes Kulovits, Michael Kraxner, Markus Plangg, Christoph Becker, Sean Bechhofer.** Lisboa, Portugal : 10th International Conference on Preservation of Digital Objects (iPres 2013), 2013.
7. **Donal Fellows, Markus Plangg.** *Design and implementation of the preservation component catalogue.* s.l. : SCAPE Project Deliverable (D7.3), 2014.
8. *Scalable preservation decisions: A controlled case study.* **Hannes Kulovits, Christoph Becker, Bjarne Andersen.** s.l. : Archiving 2013. Society for Imaging Science and Technology, 2013.
9. *Large-scale content profiling for preservation analysis.* **Petar Petrov, Christoph Becker.** s.l. : 9th International Conference on Preservation of Digital Objects (IPRES 2012), 2012.
10. SCAPE Plan Management API. *github.com/openplanets/scape-apis.* [Online] 2014. https://github.com/openplanets/scape-apis/blob/master/Plan%20Management%20API_V1.0.pdf.

5 Glossary

The following terms and abbreviations are used throughout this report:

Table 1: Terms and Definitions

Term/ Abbreviation	Description
API	Application Programming Interface
Automated Planning	A systematic and semi-automatic process that provides the ability to assess the impact of influencers and specify actionable preservation plans that define concrete courses of actions and the directives governing their execution. This is the operative management of obsolescence and maximizing expected value with minimal costs.
Automated Watch	A systematic and semi-automatic process that provides the ability to monitor external and internal entities for changes having a potential impact on preservation and to provide notification. The Automated Watch entity denotes the architectural software component that supports the Automated Watch process.
(SCAPE) Components	SCAPE Components are Taverna Components, identified by the SCAPE Preservation Components sub-project, that conform to the general SCAPE requirements for having annotation of their behaviour, inputs and outputs. SCAPE components may be stored in the SCAPE Component Catalogue.

Component Catalogue	The Component Catalogue is a searchable repository for the definitions of SCAPE Components, Component Families and Component Profiles. The Component Catalogue is implemented by the myExperiment service [20] and implements the Component Service API [36].
Component Management	Tools and the Component Catalogue Service encompassing the creation, storage and cross-organisational sharing of SCAPE Components.
Component Profile	A definition of an interface that a Component should conform to. A component profile defines what input ports and output ports the component must have, what inputs and outputs may be optionally present, and what semantic annotations may be attributed to the component and its ports.
Execution Environment	An abstract layer of the Execution Platform which provides a placeholder representing functionality to be fulfilled by a specific technology. The Execution Environment provides the physical infrastructure to perform computation. An example might be the nodes of a Hadoop cluster.
Execution Platform	An infrastructure that provides the computational resources to enact a Preservation workflow and execute Preservation actions. Abstracted into three layers: the Execution Environment; the Job Execution Service and the Job Submission Service API.
File	A file is a named and ordered sequence of bytes that is known by an operating system. A file can be zero or more bytes and has a file format, access permissions, and file system characteristics such as size and last modification date.
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
Job Execution Service	An abstract layer of the Execution Platform which provides a placeholder representing functionality to be fulfilled by a specific technology. The Job Execution Service provides job scheduling functionality, allocating computing tasks amongst the available hardware resources available within the Execution Environment. An example might be Taverna-Server or Hadoop.
Job Submission Service API	An abstract layer of the Execution Platform which provides a placeholder representing functionality to be fulfilled by a specific technology. Provides the entry point to the Execution Platform, implementing a remotely accessible interface to enable a user or client application to schedule and execute workflows (jobs) on the Execution Environment. The exact interface depends on the underlying Job Execution Service and Execution Platform, but typical examples would be the Hadoop API provided over a SSH connection, or the Taverna-Server REST API over HTTP.

OAIS	Open Archival Information System
OWL	Web Ontology Language
Preservation Plan	A preservation plan is a live document that defines a series of preservation actions to be taken by a responsible institution due to an identified risk for a set of digital objects or records (called a collection). It is defined by Plato and stored in a Plan Management Service.
Preservation Action Plan	A Preservation Action Plan is part of a Preservation Plan and describes a set of digital objects, an operation (typically a transformation) to apply to each of them, and a rule that allows the determination of whether the operation on a particular digital object was successful. Success is determined on the basis of characteristics measured on the instantiation of the digital object, what it was transformed into, or the comparison of what it was and what it became. A Preservation Action Plan does not describe how to instantiate the DO, where to archive successful transformations, or where to report the outcome of applying the Preservation Action Plan.
RDF	Resource Description Framework
REST	REpresentational State Transfer
SCAPE	Scalable Preservation Environments
SPARQL	Simple Protocol and RDF Query Language
SRU	Search/Retrieve via URL
SSH	Secure SHell
Taverna Component	Taverna components are Taverna workflow fragments that are stored independently of the workflows that they are used in, and that are semantically annotated with information about what the behaviour of the workflow fragment is. They are logically related to a programming language shared library, though the mechanisms involved differ. Taverna components are stored in a component repository. This can either be a local directory, or a remote service that supports the Taverna Component API (e.g., the SCAPE Component Catalogue). Only components that are stored in a publicly accessible service can be used by a Taverna workflow that has been sent to a system that was not originally used to create it.
Taverna Server	Taverna Server is a multi-user service that can execute Taverna workflows. Clients do not need to understand those workflows in order to execute them.

Taverna Workbench	The Taverna Workbench is a desktop application for creating, editing and executing Taverna workflows.
Taverna Workflow	A Taverna workflow is a parallel data-processing program that can be executed by Taverna Workbench or Taverna Server. It is stored as an XML file, and has a graphical rendering.
XML	eXtendable Markup Language

6 Appendix

6.1 Quality level definition as Schematron schema

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern>
    <title>QLDs based on plan iPRES 2013 DEMO(test_plan_123)</title>
    <rule
      context="measure[@type='http://purl.org/DP/quality/measures#134' and (@subject != 'inputFile')]">
      <assert test="(. != 'No')">automated QA supported must have (one) of the following values: [Yes]</assert>
    </rule>
    <rule
      context="measure[@type='http://purl.org/DP/quality/measures#123' and (@subject != 'inputFile')]">
      <assert test=" .  &lt;=  4.0">comparative file size must be less than or equal to 4.0</assert>
    </rule>
    <rule
      context="measure[@type='http://purl.org/DP/quality/measures#117' and (@subject != 'inputFile')]">
      <assert test="(. != 'lossless') and (. != 'lossy')">compression type must have (one) of the   following values: [none]</assert>
    </rule>
    <rule
      context="measure[@type='http://purl.org/DP/quality/measures#51' and (@subject != 'inputFile')]">
      <assert test="(. != 'No')">image width equal must have (one) of the   following values: [Yes]</assert>
    </rule>
    <rule
      context="measure[@type='http://purl.org/DP/quality/measures#53' and (@subject != 'inputFile')]">
      <assert test="(. != 'No')">image height equal must have (one) of the following values: [Yes]</assert>
    </rule>
    <rule
      context="measure[@type='http://purl.org/DP/quality/measures#251' and (@subject != 'inputFile')]">
      <assert test="(. != 'No')">EXIF: all tiff data retained must have (one) of   the following values: [Yes]</assert>
    </rule>
  </pattern>
</schema>
```