Goals and requirements for PT from the perspective of PW

Motivation

This section quickly outlines key goals for PW. It focuses on the *components* that are developed in WP12 and WP14 since these are the places of interaction with PC and PT.

Planning goals

- *Key goal: Create and revise preservation plans efficiently and effectively*
- Understand and document and take into account drivers, constraints, goals
- Find best available preservation actions
- Run automated planning experiments using taverna workflows
- Create executable preservation plans
- Deploy executable plans to repository easily and quickly
- Be available for public testing and usage
- Assess impact of changes in influencers

Watch goals

- Enable the planning component to automatically monitor entities and properties of interest
- Enable human users and software components to pose questions about entities and properties of interest
- Collect information from different sources through adaptors
- Act as a central place for collecting relevant knowledge that could be used to preserve an object or a collection
- Enable human users to add specific knowledge
- Notify interested agents when an important event occurs
- Act as an extensible component.

What this means for the platform: PT requirements from PW

Goal 1: Be able to run large-scale deep content characterisation

Planning and Watch need to identify issues and preservation risks in content. Watch needs to run longitudinal analysis on large collections to identify trends and estimate lifelines of formats and features over time. Finally, we need to be able to monitor evolution of content over time. This requires us to be able to run in-depth characterisation on large-scale collections and collect the measures in a well-defined vocabulary.

- i.e. extract information from all objects
  - ideally using something like fits that delivers in-depth information
- i.e. also: provide information on each/all objects upon request

What happened so far?

IM and SB are running FITS on their collections, it takes quite long. TUW has developed a variant of FITS that is ~10x faster than the original on web archives. Parallelisation is semi-manual as of yet. Shared vocabulary is based on fits so far.

Goal 2: Provide discovery of applicable actions at any time

In Planning and Watch, we are searching for the best way to treat an identified issue. This requires us to discover potential ways of treatment, i.e. actions, based on search criteria:

- input objects (format...)
- output
- environment information
- standard SQUARE attributes
- business factors
    - license
    - cost factors
- (...to be completed. Longer discussion in D14.1!)

What is an action? Our working understanding is that an action is anything (be it a Java component or a Unix command line tool or a Beanshell script) in a workflow that is considered a relevant preservation action by someone. This probably includes at a minimum

- all migration components that are developed by PC
- all migration workflows published on myexperiment?
- ....what else?

Rationale: When looking for an action, you don't want to constrain how a possible action comes into being, you look at an action fulfilling requirements. On the one hand, actions can be easily chained into compound actions (think .tex → .ps → pdf, which becomes the action of converting LateX into PDF); on the other hand, any of these could be *implemented as* any kind of workflow.

What happened so far?

We have a standard model for action properties; action services is using a subset of these, but yet without URIs. Environment information may be missing at the moment from this model.

Goal 3: Provide discovery of applicable characterisation and QA components according to search criteria

In order to (dynamically?) construct planning experiment workflows and executable plans, planning needs to find out which components can measure certain things, e.g. which component is available to measure the equality of pagecount of a .doc and its converted .pdf representation; or simply identify the puid of a converted object to verify the format conforms to what the migration action claimed to produce.

Search criteria therefore are (at a minimum)

- input format
- properties to measure
- (output format for QA)
- additional (standardised) attributes, such as average runtime (e.g. to enable tradeoff and optimisation for executable plan specification)
- ... all according to a standardised vocabulary

What happened so far?

Standard properties have been defined (see D14.1) for QA, but are not yet used by QA and/or CC. To our knowledge, CC+QA components are not yet systematically published and discoverable in workflows.

Goal 4: Provide unified interfaces to any component and standardised results

Each standard class of component (action, characterisation, QA) needs to be invoked in the same way, independent of whether it is e.g. FITS, JHOVE2 or something else. All results need to collected in a well-defined vocabulary (!!), e.g. the output ports of all workflows need to be properly named and adhering to standard vocabulary.

Goal 5: Ensure consistent execution across environments

In order to ensure consistency between experiments and large-scale operations, the execution of each component and/or workflow must always deliver the same functional results independent of the deployment mode (e.g. independent of single- or multi-node execution, independent of the parallelisation strategy etc.)

Goal 6: Provide environment to test applicable actions on real data

At the heart of planning, a planning experiment collects relevant evidence by creating and running an experiment on sample content. Thus, we need support to

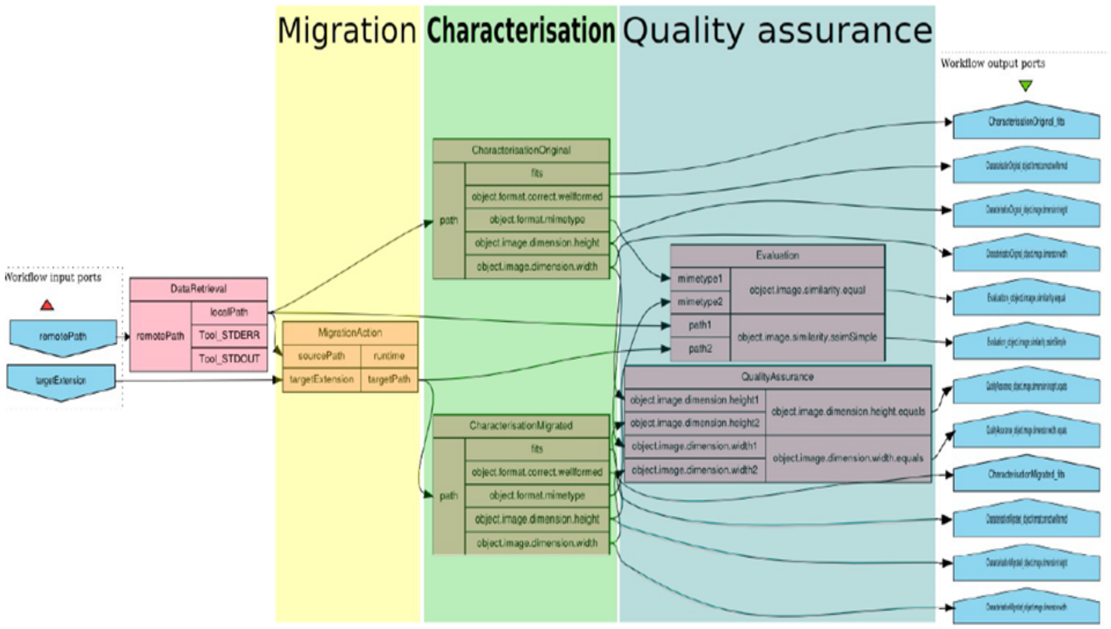
- allow on-line invocation of preservation actions on input data provided by PW and
- dynamically execute workflows at any time.

That means that there needs to be a deployment, it needs to be always on and available, and it needs to be (quite) up-to-date. However, it does not need to be a cluster (parallelisation not needed) since planning experiments are normally not large-scale!
What happened so far?
The central instance is running at IMF and accessible to SCAPE partners. AIT runs a cluster that can be made available to SCAPE partners. Since planning experiments run on very few representative samples, scalability is not a issue and the disadvantages due to the overhead outweighs the advantages.
Taverna has a server version that supports upload and invocation of workflows via a REST interface. However this would have to be maintained in parallel to the platform. Also running the experiments on the same platform as the executable plan increases the confidence in the experiments.
It has to be ensured that this solution works also with future releases of the platform.
After discussion with Rainer, the currently preferred solution is running a single-node instance of the cluster dedicated for planning experiments. The cluster nodes already consist of VMs that are maintained for the platform.
There are two options where to deploy the VM:

- The VM is deployed at AIT and continuously available to the planning component. This bring transmission overhead for experiments on large input files.
- AIT provides a VM Image to deploy on a server at TUW. While AIT uses xen, TUW uses ESX but the VM images can either be converted or xen can be run inside an ESX VM.

Goal 7: Provide "planning experiment workflows"

To evaluate preservation actions, experiments are conducted on the sample content. These workflows combine actions, characterisation and QA (and potentially supportive components) into a fairly standardised evaluation workflow. An example is shown below.


Migration
Characterisation
Quality assurance
Workflow input ports
remotePath
targetExtension
DataRetrieval
remotePath
localPath
Tool_STDERR
Tool_STDOUT
MigrationAction
sourcePath
runtime
targetExtension
targetPath
CharacterisationOriginal
path
fits
object.format.correct.wellformed
object.format.mimetype
object.image.dimension.height
object.image.dimension.width
CharacterisationMigrated
path
fits
object.format.correct.wellformed
object.format.mimetype
object.image.dimension.height
object.image.dimension.width
Evaluation
mimetype1
mimetype2
path1
path2
object.image.similarity.equal
object.image.similarity.ssimSimple
QualityAssurance
object.image.dimension.height1
object.image.dimension.height2
object.image.dimension.width1
object.image.dimension.width2
object.image.dimension.height.equals
object.image.dimension.width.equals
Workflow output ports
CharacterisationOriginal_fits
CharacterisationMigrated_fits

These workflows combine actions with characterisation and QA. They need to provide the results as standardised measures in an agreed vocabulary so they can be used in planning experiments for evaluating actions. These can be developed and published by TUW. Their successful execution requires all of the above provisions!

What happened so far?

We have created such workflows to analyse the standard building blocks required. Possible options for developing and delivering them are outlined in the next section.

Goal 8: Provide a facility for deploying and executing plans

Once a choice is made in planning for the recommended action, an executable plan is created. This includes a workflow based on the workflow above, but with a number of differences.
First, the workflow may have reduced characterisation and QA. Additionally, the executable plan specifies the content set on which it shall be executed.
Finally, it will specify acceptable output ranges for all measures, which means ... TBC

Thus, we need a facility to operationally deploy and execute plans:

- provide interface for deploying plans for execution
- execute plans on specified set of content according to data connector API
- provide reporting according to plan specs and reporting API

What happened so far?

PW.WP.3 and PT.WP.5 are jointly discussing requirements and interfaces for plan management and deployment.

Goal 9: Ensure timely availability of SCAPE components for planning experiments

To make sure that ongoing development effort of component developers (PC) are leveraged, we need to ensure they have an easy way of sharing results and that these are indeed fed into the appropriate catalogues/registries so that PW and PT know about them and can query that at

any time. That means that there needs to be an easy way for component developers to publish their results into appropriate catalogues or registries. This allows the planning component to discover components and provided these as options to the user and watch to notify interested parties of new components. Additionally it allows PT to deploy the components and make them available for planning experiments, executable plans and other SCAPE partners.
We have to deal with the tradeoff between fast and easy publishing on the one hand and the discoverability, metadata quality etc on the other hand.

Overview of Ideas for different Workflow models

To enable the planning component to run automated planning experiments using taverna workflows planning needs a way to discover components, create taverna workflows from them and run them in a timely fashion. Additionally executable plans have to be created based on these experiments workflows.
A typical workflow contains components for migration actions, characterisation of both the original and the migrated object as well as QA. These three component types typically have different levels of interface complexity which plays a role in how generic the workflows can be created. In general, many workflows for planning experiment will look like the one pictured above (see Goal 7).

The configuration of a migration action contains the tool to use and parameters but can also consist of a series of migration steps (e.g. .tex → .ps → .pdf) wrapped in a workflow. It has the original object as input parameter and the migrated object as output parameter.
A characterisation component takes an object (original or migrated) as input but has a number of output parameters depending on the used tools and parameters. Of course the tools could also be wrapped in workflows (e.g. for postprocessing of the characterisation results).
QA components are typically the most complex of these three component types in terms of interface and the components themselves. Depending on the type of QA, the component need the objects themselves and different results from characterisation. They also provided a wide array of output.
With this in mind it may make sense to use different approaches for different component types.

In this section, we quickly outline possible options for modelling, developing and maintaining sets of experiment workflows that can be used in planning to evaluate preservation action components (and workflows) using characterisation and QA components/workflows.
Note that these options do not have to be exclusive, i.e. it may be best to combine them and/or to use a phased approach.

Generic workflow with smart tool(wrapper)s

Use a generic workflow independent of content type. For each typical migration step (ie.e Migration action, characterisation, QA) create a wrapper for the corresponding components that calls the components based on input and output parameters. All logic for selecting components is in this wrapper. Additionally the wrapper must be able to call components independent of the component type (e.g. command line tool, workflow, web service).
The workflow only passes data to and between the wrappers and collects the output of all wrappers. Thus there is almost no logic in the workflow.

Generic workflow with component IDs as runtime parameter

Use a generic workflow independent of content type. For each typical migration step (i.e. Migration action, characterisation, QA) a list of components IDs and their corresponding output is provided on an input port. A migration step consists of a taverna service, that understands the provided component IDs and dynamically calls this component with all necessary parameters.
The logic for selection of components is done before calling the workflow. The workflow only passes data to and between the taverna services that call the components and collects the output data. Thus there is almost no logic in the workflow.
The taverna service must know how to call the component, independent of the component type.

Comprehensive workflow with agnostic tools

Use multiple workflows that contain components for similar preservation actions (e.g. migration by content type, filetype or file group). A typical workflow for image migration contains a sensible selection of migration actions, characterisation and quality assurance components readily linked together.
Logic to choose which workflow to use is external before running the workflow. The workflow itself is adapted to the components it contains.
The workflow could always run all components or have an option to activate/deactivate components via an input port. This requires conditional branching which is not directly supported by Taverna but available with a workaround#.

User generated workflows

Domain experts generate complete migration workflows (including migration action, characterisation, QA) that abide by a standard interface so they can be called by the planning component automatically. The workflows could be based on templates that are either predefined or generated according to the decision criteria available in the planning component. The workflows can be uploaded to myexperiment, tagged and possibly annotated with other metadata. This would allow others to use them directly or adapt them to their needs.
Workflows contain all logic on which components to use (supported by templates).
Q: How can you test/validate compatibility and conformance to the interface so that Plato can really call it?
Q: If someone locally develops a workflow, (how) does this workflow get executed in the platform/on the testing environment? (Idea: upload to plato; plato provides http link to taverna environment; non-standard / non-SCAPE workflow elements are not supported. Is this workable and sufficient?)

Programmatically generated workflow

For each plan a workflow is programmatically generated with all previously selected components and the necessary links between them. All logic on which components to select and how to link them is external but encoded in the workflow.
CB: I think this can only be considered seriously as an option to create the entire workflow if we have reviewed a predecessor demonstration of the feasibility of this generation, including examples. Otherwise the risk seems very high.
But we may be able to generate stubs or templates that then need to be completed by the user, e.g. to make sure the shared vocabularies are used for the port names.

Workflow template where components are replaced programmatically

A generic generic workflow template is created with all typical migration steps (migration action, characterisation, QA). The workflow contains dummy workflow services that are replaced programmatically with components (e.g by changing the t2flow XML). The workflow does not contain any logic. Which component to select is done externally before adapting the workflow. The replacement mechanism must support all component types (e.g. command line tools, workflows, web servies).