




# Identification of triggers and preservation Watch component architecture, subcomponents and data model

## Authors

Kresimir Duretec, Petar Petrov, Christoph Becker (Vienna University of Technology)  
Luis Faria (KEEP Solutions)

January 2012

*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* 

## Table of Contents

1	Introduction.....	7
1.1	Planning and Watch.....	7
1.2	State of the art and current practice.....	10
1.2.1	Monitoring activities in Business Management.....	10
1.2.2	Watch in Digital Preservation.....	11
1.2.3	Supporting tools and services.....	13
1.2.4	Current practice.....	15
1.3	Goals.....	17
1.4	Concepts and terminology.....	18
2	Requirements.....	22
2.1	Significant events and triggers.....	22
2.2	A planning-and-watch scenario.....	26
2.2.1	The British Library Newspapers plan.....	27
2.3	Policy-driven Watch.....	32
2.4	Sources of information.....	32
2.4.1	Format registry and component catalogue.....	33
2.4.2	Policy model/Policy Catalogue.....	33
2.4.3	Repository.....	33
2.4.4	Experiments and results.....	34
2.4.5	Content profile.....	34
2.4.6	Web browser snapshots.....	35
2.4.7	Simulator.....	35
2.4.8	Human knowledge.....	36
2.5	Significant Events and Questions.....	36
3	Architecture.....	37
3.1	Data Layer.....	40
3.1.1	Knowledge Base.....	40
3.1.2	Watch Request.....	43
3.1.3	Putting it all together.....	48
3.2	Application Layer.....	48
3.2.1	Watch Central Service.....	48
3.2.2	Assessment Service.....	49
3.2.3	Notification Service.....	50
3.2.4	Monitoring Service.....	50

3.2.5	Source Adaptors .....	51
3.2.6	Web API .....	52
3.3	Front End Layer.....	52
4	Technology Discussion.....	54
4.1	Data Layer .....	54
4.2	Volume estimation .....	55
4.3	Application Layer .....	58
4.4	Front End Layer.....	59
5	Risks.....	59
6	Summary and Outlook.....	60
6.1	Summary .....	60
6.2	Next Steps.....	61
7	References.....	62

## Executive Summary

The SCAPE project is set to advance the planning and control of preservation operations from ad-hoc decision making to a continuous management activity. This document lays out a key component for this improvement. In Automated Watch, the project will provide automated mechanisms to support the monitoring and evolution of preservation plans over the lifecycle of digital content and react to a dynamically changing environment and user behaviour. Plan enactment and continuous operations need to be monitored. The collected measurements need to be analysed automatically to trigger appropriate events. Finally, critical changes in the technological environment should be detected and lead to automated notifications that can trigger decision making.

The work package Automated Watch will develop mechanisms for monitoring content, designated user communities and systems in order to support policy-driven semi-automated reactions to identified conditions. The work package will model and implement watch mechanisms, triggers, and suitable actions to be taken for each trigger. This will support closed-loop preservation processes in which automated monitoring of collections, actions, plans, systems, and the environment triggers appropriate diagnosis and reaction. The first milestone towards this integrated vision of preservation operations is the design of an automated Watch component. The component will collect information from a variety of sources and monitor the gathered evidence for properties and trends of interest.

The key goals of the Watch component are outlined as follows.

1. Enable the planning component to automatically monitor entities and properties of interest
2. Enable human users and software components to pose questions about entities and properties of interest
3. Collect information from different sources through adaptors

4. Act as a central place for collecting relevant knowledge that could be used to preserve an object or a collection
5. Enable human users to add specific knowledge
6. Notify interested agents when an important event occurs
7. Act as an extensible component

Based on this goal-oriented view, this report analyses the state of the art and clarifies terminology. It describes and classifies events of interests. An analysis of drivers that need to be monitored allows us to describe the range of triggers that should lead to preservation planning and action activities. These triggers can be classified according to their measurement sources and according to which of these refer to monitoring compliance, risks and opportunities. Essentially, on completion and deployment of an operational preservation plan that specifies preservation actions and associated characterisation, quality assurance and reporting activities, we need to monitor compliance of the operations to the specified tolerances as well as risks and opportunities associated with chosen and potential activities.

This report outlines the key necessary components that comprise what in the SCAPE proposal is referred to as “The Watch Service”. The Watch Service in this sense is what should actually be called the “Watch component,” which is what is defined in this document. The Watch component is comprised of the Watch Service, a central reference point handling questions about the world and collecting the knowledge to answer them, a number of Monitoring Services that can monitor certain aspects of the world, and a number of adaptors providing specific connectors to information sources that represent the world.

The key sources of information that are currently considered are: format registries and the SCAPE preservation component catalogue; policy models; repositories; experiment results; content profiles; human knowledge; and two specific modules to be developed within Watch (a snapshot service comparing web page renderings to baseline rendering snapshots, and a simulator for assessing the effects of planning and watch decisions). These information sources will be fed into a Linked Data knowledge base through a number of adaptors. Configurable conditions and rules will be provided to create flexible notification mechanisms for subscribers. These building blocks together address the internal and external monitoring capabilities that an organization needs to successfully manage digital preservation operations over time.

# 1 Introduction

This report presents the high-level motivation and vision for SCAPE Automated Watch and describes the design of the automated Watch component to be developed.

A key goal in the SCAPE project is to provide automated mechanisms to support the monitoring and evolution of preservation plans over the lifecycle of digital content and react to a dynamically changing environment and user behaviour. Plan enactment and continuous operations need to be monitored. The collected measurements need to be analysed automatically to trigger appropriate events. Finally, critical changes in the technological environment should be detected and lead to automated notifications that can trigger decision making.

The work package *Automated Watch* will develop mechanisms for monitoring content, designated user communities and systems in order to support policy-driven semi-automated reactions to identified conditions. The work package will model and implement watch mechanisms, triggers, and suitable actions to be taken for each trigger. This will support closed-loop preservation processes in which automated monitoring of collections, actions, plans, systems, and the environment triggers appropriate diagnosis and reaction.

The first milestone towards this integrated vision of preservation operations is the design of an automated Watch component. The component will collect information from a variety of sources and monitor the gathered evidence for properties and trends of interest. It will be an extensible platform that can accommodate a wide range of concerns of interest.

This report is structured as follows. The first section presents the key goals, concepts and terms that are the basis of Automated Watch. Section 2 discusses the functional requirements of the Watch component. Section 3 provides an overview of the architecture of the component and discusses architectural details and design. In section 4 we provide a technical discussion about the technology stack that will be used during the development phase. We discuss key technological aspects on the data layer and the application and front-end layers. Section 5 provides a brief risk analysis of the important aspects in the component and section 6 offers a brief summary and outlook.

## 1.1 Planning and Watch

The work presented here strongly builds on the preservation planning framework and tool **Plato** (Becker et al., 2009). As shown conceptually in Figure 1, the decision-making phases in Preservation Planning (Define Requirements; Evaluate Alternatives; Analyse Results; Build Preservation Plan) are followed by a *Monitor* phase that may lead to a revision in a plan, triggering a new iteration in a plan's lifecycle. The planning component Plato delivers the basic decision making capability, but does not explicitly address monitoring. It will provide key input to monitoring and support the assessment of changes, but does not currently implement monitoring functionality.

While the understanding of the concepts relevant to planning (objects, technology, etc.) has progressed considerably since the creation of this diagram, the key framework is still valid, and we see that we need to complement the preservation planning, i.e. decision making method, with a continuous monitoring ability.

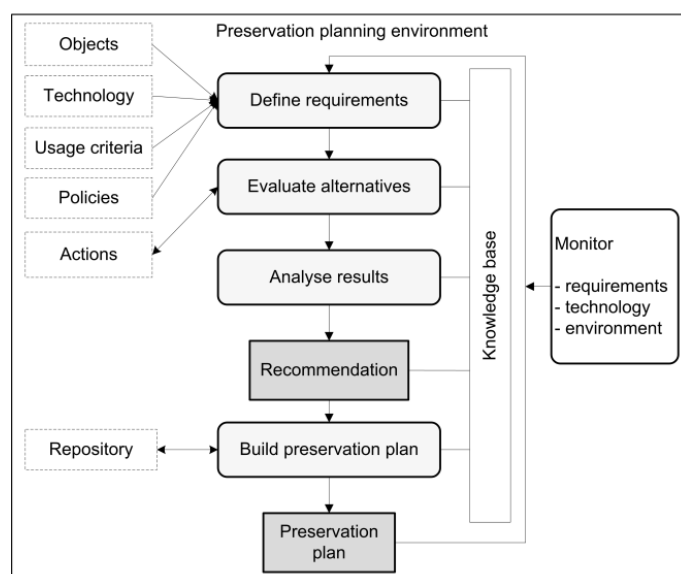


Figure 1 - Preservation Planning cycle (Becker et al., 2009)

The capability<sup>1</sup> that is required to achieve this monitoring in an organizational context has been specified in the SHAMAN Reference Architecture (RA) (Antunes, Becker, Borbinha, Proença, & Vieira, 2011) as the second major Planning capability (besides Planning Operational Preservation): Monitoring.

The SHAMAN RA defines it as follows:

*The ability to monitor operations (including in particular the execution of plans) and the environment, i.e. the ability to monitor all influencers having a potential impact on plans to ensure conformance of results to expected outcomes and notify the decision making capability (Planning Operational Preservation) of a change that requires assessment.*

This essentially requires monitoring of operations (internal to the system and organization), and outward monitoring of the world and its properties of interest (external to the system).<sup>2</sup>

The SHAMAN RA correspondingly specifies two types of monitoring that together fulfil the Monitoring capability:

*Internal Monitoring is the ability to monitor operations for certain properties of interest, which include operations specified by plans and operational attributes of the system, i.e. internal influencers. Internal influencers of interest include (but are not necessarily limited to):*

<sup>1</sup>A capability is “An ability that an organization, person, or system possesses. Capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve.” [http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html#tag\\_03\\_30](http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html#tag_03_30)

<sup>2</sup>This very wide scope (“the world”) is chosen on purpose to reflect the fact that a priori, almost any condition in the widest environment of an organisation can have an ultimate impact on decisions. Of course, any concrete monitoring activity will have a particular specified scope of interest.

- *Operational statistics about the execution of delivered plans*
- *Operational statistics about content dissemination*
- *User feedback about satisfaction with respect to user access requirements*
- *User feedback about satisfaction with respect to content requirements*

*External Monitoring is the ability to monitor external influencers of interest. These influencers include technological opportunities; technology correspondence as an indicator of impending obsolescence and misalignment to user access requirements; changes in the user community; shifts in producers' technology that lead to different properties of content deposited into the organization; change in access requirements; changes in costs of used technology that need to be considered in operations; and regulations.*

The term Watch was coined in Planets (PLANETS, 2009; PLANETS, 2010). In this view, Watch encompasses monitoring of the repository administration, the user community, the organisation, the producers, and the technical environment. Additionally, the Watch function was modelled to receive executed preservation plans and feedback. The function was assumed to be partitioned into Monitoring, Risk Analysis, Representation Information Update, and a Testbed. The latter was defined as a controlled environment for conducting experiments of tools on content.

In SCAPE, the “testbeds” are the real-world environments that provide the scenarios for specific problems to be addressed, so that the project can validate its outcomes. The controlled environment is presented by the Platform and accessed by the planning tool. It produces experiment data that are a valuable source of information.

In SCAPE, we merge the viewpoint of the *Planets functional model* with the capability-oriented view of SHAMAN that clearly separates decision making from information gathering. We furthermore aim at supporting organisational capability development through flexible and extensible services that support the automated elements of preservation management without constraining organisations to follow a specific model in their deployment of the planning capabilities. Thus, the *automated Watch component* supports as far as possible the automation of the Monitoring capabilities. It provides a Watch Central Service frontend that manages aspects to be monitored and uses a number of Monitoring Services to gather information for analysis and reasoning.

Essentially, the *usage* of the Watch service can be reduced to the following steps:

1. An agent has a question about a certain aspect of the world that is of interest to the agent. This interest is driven either by a plan, a policy, or another type of concern.
2. The agent expresses this interest in the form of a question about a property that represents this interest.
3. The function of Watch then is to find a method (and possibly develop a tool) to deliver an answer to this question that is correct, timely, and reliable.
4. Having received an answer to the question, the agent will want to assess the meaning and impact of this answer. This may require consultation of the decision-making capability.

For the recipient, an answer may have a variety of impacts, and it may in particular overlap with other answers. There may be simple conditions attached to certain questions that trigger an event when these conditions are met (e.g. when the answer changes by more than 5%). It is not, however, possible for the automated watch process to assess the *cumulative* impact of multiple answers to an

external entity consuming the answers, as this would require much more than the knowledge that is available. Essentially, this means that the Watch component itself should be ignorant of the ultimate effects of changes, and just raise events to be assessed. The primary purpose of the Watch component is to make the state of the world available for assessment, not to assess it.

In the case where the question was motivated by an existing plan (or several plans), this significance assessment relies on the knowledge intrinsic to the originating preservation plan(s). It can thus request an assessment by the planning component. In the case of a question that originated from a policy, local conditions should be included that can be composed of multiple single conditions, but evaluated within Watch. The default reaction will then be a planning iteration.

For the development of the monitoring capability during SCAPE, this means that the following steps are required:

1. High-level architecture and concepts
  - a. Development of a generic model of the world and its properties of interest, based on references, User Group feedback and Tested scenarios
  - b. Clarification of terminology
  - c. Identification and classification of watch events/triggers/questions according to information sources, conditions, possible impact and actions, and derivation of monitoring goals and objectives from reference models
  - d. Vision for a high-level architecture that supports the dynamic extension of watch (answering) services (key components and interfaces)
  - e. Clarification of the decoupled interplay between these
2. Design of the software architecture
  - a. Revision of the generic model of the world and its properties
  - b. Specification of components required and external dependencies (e.g. rule manager, plugin manager, repository adaptor, assessment, trigger execution)
3. Design and development
  - a. Of the core architecture components
  - b. Of some services where basic tools exist, to validate the architecture
4. Preliminary assessment of the properties/questions of interest according to
  - a. Importance
  - b. Feasibility and effort of developing an (automated) answer
5. Based on the previous steps: Derivation of a roadmap for the development of services, based on an assessment of feasibility, criticality, timelines and available resources with expertise and skills set
6. Development of the services, simulation and testing, further refinement of architecture and services, etc.

This document addresses the first three aspects and thus sets the basis for successfully developing the software systems of Watch in SCAPE.

## **1.2 State of the art and current practice**

### **1.2.1 Monitoring activities in Business Management**

Watch, in the sense of monitoring the surrounding environment, is a very common subject in various domains. In the domain of management science, Stafford Beer defines that any viable or autonomous system must be able to cope with the demands of the changing environment. This adaptability is



what ensures its odds of survival and allows a system to be more perdurable. Beer defines that there is a major component of input to top-level decisions: information about the environment set by the outside world. This component is responsible for looking outwards to the environment to monitor how the system needs to adapt to remain viable (Beer, 1981). Beer further defines that internally, in another component, there is an internal representation of the real world and that the system constantly tries to match the information that comes from the monitoring activity with this model of the world. Beer also defines that we can consider the system as adaptive where the re-investment (on a business decision or plan) is fed back into the process creating a loop, and that the monitoring component must be able to track the performance of the plan so it may be continuously adapted.

From a business and economic perspective, knowledge about the environment is also seen as a primary capability needed for survival and the drive for success. "Today, knowledge in all its forms plays a crucial role in economic processes" (OECD, 1996). This applies to nations as well as to businesses and is essential whenever long-term survival is a goal. The need to define and certify the processes that relate to knowledge led to the creation of standards that define Research, Development and Innovation (RDI) management systems, like the Portuguese standard for Research, Development and Innovation Management (Portuguese Institute for Quality, 2007). This Innovation standard defines three interfaces with the outside world:

- **Technological interface** - aims to support technological watch, cooperation and prevision to interact with external technological and scientific research. The technological watch is defined as the systematic, structured and organized gathering of knowledge about economical, technological, social and commercial developments;
- **Market interface** - aims to use marketing and client feedback to gather knowledge on existing markets, requirements, drivers and preferences;
- **Organization interface** - aims to stimulate, gather and manage internal knowledge and creativity.

### 1.2.2 Watch in Digital Preservation

In digital preservation, the focus on maintaining a system viable and ensure long-term survival is even more important. In the functional model of OAIS, the monitoring functions are part of the Preservation Planning function and can be divided in:

- **Monitor Designated Community** - "interact with Consumers and Producers to track changes to their service requirements and available product technologies" (CCSDS, 2002);
- **Monitor Technology** - track "emerging digital technologies, information standards and computing platforms (i.e. hardware and software) to identify technologies which could cause obsolescence in the archive's computing environment and prevent access" to content (CCSDS, 2002).

In 2009, the Planets Functional Model (Sierman & Wheatley, 2009) described a model for preservation planning and watch, depicted in Figure 2. This model does not try to substitute the OAIS, but to refine and extend it to the needs of the Planets project. An cross-evaluation between the models was later described in (Sierman & Wheatley, 2010) where the entities of both models were mapped, producing a series of recommendations for OAIS.

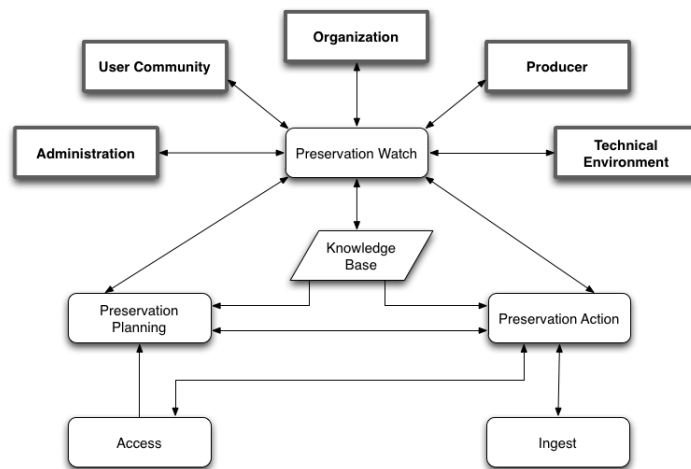


Figure 2 - Planets Functional Model

In the Planets Functional Model, the two OAIS monitoring functions were combined into *Preservation Watch* function. Also, a *Knowledge Base* component is made relevant, and is defined as "an umbrella term for a repository of a variety of key information which will inform preservation processes conducted within the Digital repository" (Sierman & Wheatley, 2009). This Knowledge Base loosely maps to the *Representation Network* concept described in the OAIS Model and is also analogous to the Beer's internal model of the world concept. The model defines that Preservation Watch is composed by the sub-functions:

- **Monitor** - collecting preservation related information from a variety of internal and external sources
- **Risk Analysis** - assessment of the collected information identifying critical risks that are then relayed to Preservation Planning
- **Representation Information Update** - updates the Knowledge Base with new collected information
- **Testbed** - controlled environment to execute experiments to better assess preservation tools and services

The Planets Functional model further describes which entities should be monitored:

- **Administration** - monitor content collection and usage profiles;
- **Organization** - monitor internal changes in the organization that might affect the ability or strategy to preserve the content;
- **Knowledge Base** - monitor the information in the (internal or external) Knowledge Base and assess if a risk arises;
- **User Community** - monitor user community expectations and scope;
- **Producer** - monitor changes in producer technologies and scope;
- **Technical Environment** - monitor developments in technology that might lead to risks or opportunities.

Recently, a new reference architecture emerged. In the SHAMAN project (Antunes et al., 2011), Enterprise Architecture frameworks are used to define a reference architecture for digital preservation that has a broader view of the problem and is well founded on System Architecture and Information Systems design principles. One of the premises is that OAIS has a restrictive view of the system, confining the perspective to the archive, and that the digital preservation problem extends to the whole lifecycle of an object. The other observation is a lack of systematic architecture principles and coherence in the frameworks adopted across the DP community.

The SHAMAN project defines a capability-centred Reference Architecture. In the capability model (Antunes et al., 2011), *Preserve Contents*, defined as "the ability to maintain content authentic and understandable to the defined user community over time and assure its provenance", is composed by the *Preservation Planning* and *Preservation Operation* capabilities. *Preservation Operation* relates to the preservation plan deployment and execution. *Preservation Planning* is the "ability to monitor, steer and control the preservation operation of content so that goals of accessibility, authenticity, usability and understandability are met with minimal operational costs and maximal (expected) content value" (Becker, Antunes, Barateiro, Vieira, & Borbinha, 2011). This means that *Monitoring* is one of the main capabilities of *Preservation Planning*, defined by the ability to look towards inside, to monitor operations specified by plans and the properties of the system where they run (internal monitoring), and the ability to look towards outside, to monitor external influencers that might implicate a reevaluation of the plans.

### 1.2.3 Supporting tools and services

Currently, the tools to support watch are mainly manual and incomplete. They come in the form of scientific studies and technical reports, format and tool registers and generic catalogues. Examples of these tools are (Ferreira, 2009):

- **Risk Management of Digital Information: A File Format Investigation** is a report on the impact assessment on file format migration of various digital object types, identifying the risks that executing or postponing a migration might introduce (Lawrence, Kehoe, Rieger, Walters, & Kenney, 2000);
- **INFORM methodology** – a model to predict file format obsolescence by discovering threats to preservation and their possible impact on preservation decisions (Stanescu, 2005);
- **DigiCULT<sup>3</sup> and Digital Preservation Coalition<sup>4</sup> periodic reports** – reports about technology usage trends. The main goal of these reports is not format obsolescence alert, but can still be a good basis to predict obsolescence by trend shifts;
- **File Format and Tools registries** – online registries with technical information about file formats, software products and other technical components relevant to preservation, like PRONOM<sup>5</sup>, Global Digital Format Registry<sup>6</sup> (GDFR), P2 registry<sup>7</sup>, Conversion Software Registry<sup>8</sup>, or the Open Planets Foundation registry in the works (Roberts & Swirrl IT Ltd, 2011). Unfortunately, these online registries are not yet functioning or are not very complete.

---

<sup>3</sup> <http://www.digicult.info/pages/techwatch.php>

<sup>4</sup> <http://dpconline.org/advice/technology-watch-reports>

<sup>5</sup> <http://www.nationalarchives.gov.uk/PRONOM/>

<sup>6</sup> <http://www.gdfr.info>

<sup>7</sup> <http://p2-registry.ecs.soton.ac.uk>

<sup>8</sup> <http://isda.ncsa.uiuc.edu/NARA/CSR>

- **Online software versioning catalogues** – there are many online sites that monitor new versions of software for a generic domain use. These sites do not care about digital preservation, but are very complete and commonly have a social component that can contain interesting information. Some examples of these sites are CNET’s download.com<sup>9</sup> and iUseThis<sup>10</sup>. App stores like Apple’s Mac App Store and Ubuntu’s Software Center and repositories can also be a good source of information.

None of these tools allow monitoring and watch to be done automatically, alerting the user when a preservation risk was identified. This step towards automation is very important because, as the repository content becomes increasingly more heterogeneous, the technologies that need to be watched multiply and grow beyond of what is manually feasible.

To tackle this problem, the National Library of Australia together with the Australian Partnership for Sustainable Repositories (APSR) started an initiative to create an Automatic Obsolesce Notification Service (AONS) that would provide a service for users to automatically monitor the status of file formats in their repositories against generic format risks gathered in external registries and receive notifications (Pearson & Walker, 2007).

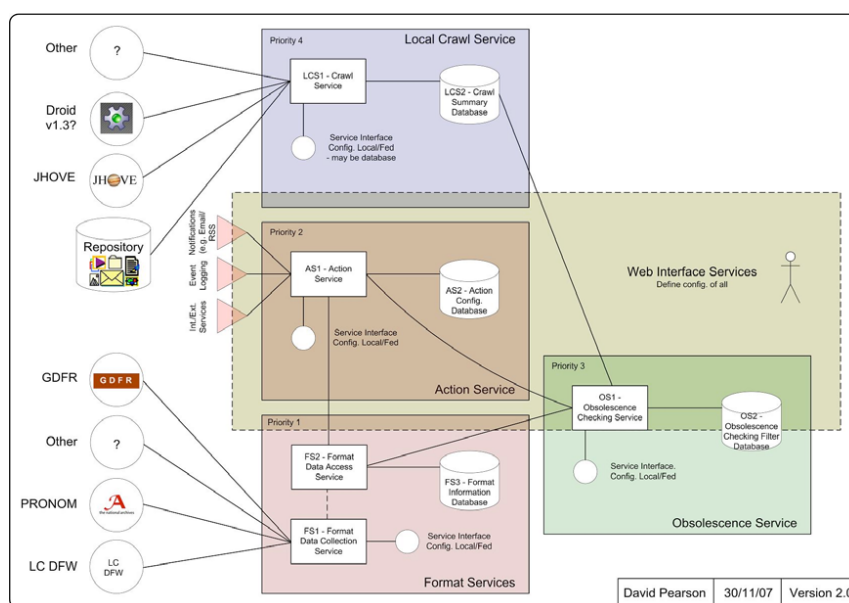


Figure 3 - AONS II Local/Enterprise Deployment Mode (Pearson & Walker, 2007)

The AONS tool was developed in two sequential projects, AONS I and AONS II, with the final goals of creating software that:

- Has support for federated and local environments;
- Is open-source, modular, platform-independent, interoperable and standards-based;
- Has a REST-ful approach;
- Has a design that decouples the core functionality from the repository and registry adapters;

<sup>9</sup> <http://download.com>

<sup>10</sup> <http://iusesthis.com>

- Is demonstrable<sup>11</sup>

The system, depicted in Figure 3, would gather collection profiles from repositories, by using format identification tools on content, and seek obsolescence risk indicators on file format information in external registries (like PRONOM). The system would also allow caching and extending format registries and the creation of new adaptors for new registries. The notification service allows subscription of various events, like end of a repository crawl or change in the information about a format, and send a notification via email, RSS feed and task boxes on the GUI.

AONS failed to achieve wide adoption primarily because of its dependence on registries and their ability to provide format obsolescence risk information on a wide range of file formats. On the development timeline this was a fair assumption as it even was the mission defined by the UK National Archives for PRONOM. But the experience on AONS revealed that registries did not give enough risk assessment information and when it was available it was "not sufficiently structured to be useful in a system-automated context without considerable human intervention" (Pearson & Walker, 2007).

AONS was limited to gathering information only about file formats, assuming that all other aspects of the world that would be relevant for digital preservation would be gathered, assessed, and represented in a structured way by the external registries. This assumption and the lack of available information in format registries constrained the usefulness of the system. Moreover, not all desired features defined in the concepts could be successfully completed.

#### 1.2.4 Current practice

During the research leading to this deliverable, a survey was conducted to assess the current practice of monitoring and watch activities within a relevant and representative group of institutions. The survey was composed of two questions:

1. What and how do you monitor? Which sources of information, on which schedule or triggered by which events, and what information do you gather from:
  - a. **Administration:** internal monitoring of collections profiles and other repository system properties,
  - b. **Organization:** changes in your own organization or other organizational aspects,
  - c. **Knowledge base:** monitoring internal and external registries of information like file formats, significant properties, earlier identified risks or even experiments with tools and services,
  - d. **User community:** monitoring and feedback from the user community,
  - e. **Producer:** monitoring of the organizations depositing material into the digital archive,
  - f. **Technical environment:** monitoring developments in the technical area that might lead to new file formats, new digital object formats or publishing methods.
2. How do you perform Risk Analysis and examples of risks identified from the gathered information?

The group of institutions that responded were:

- **British Library** - the United Kingdom National Library, represented by Paul Wheatley
- **Koninklijke Bibliotheek** - the Dutch National Library, represented by Barbara Sierman

---

<sup>11</sup> Source code available at <http://aons.sourceforge.net/>

- **Statsbiblioteket** - the Denmark National Library, represented by Bjarne Andersen
- **Science and Technology Facilities Council** - United Kingdom institution that tackles scientific research and research datasets, represented by Brian Matthews
- **Internet Memory Foundation** - international institution that tackles the preservation of internet content, represented by Leïla Medjkoune

On the majority of the answers, monitoring was based on manual labour<sup>12</sup> and many times even inexistent. In the cases where monitoring responsibilities were defined, they were scattered throughout various departments and no centralized entity was defined to receive alerts and have a complete view of the information. Table 1 summarises the answers to the first question, dividing monitoring entities as defined by the Planets Functional Model and further dividing it into sub-entities when needed. For each entity there are related a list of monitoring actions divided by the methodology used, the sources of information, and the scheduling of the monitoring.

Of the queried institutions, 40% do a formal risk assessment on their collections, but do so in a highly manual fashion. The remaining 60% do risk assessment in a very informal and ad-hoc way. It is performed by the IT department without a predefined methodology or limited to data loss and corruption. In the case of the Internet Memory Foundation, as they are not the final holders responsible for the content they harvest, they delegate risk assessment to the clients.

Table 1 - Monitoring practice summary

Entity	Practice
Administration	Monitor system with specific monitoring tools and react on event
	Regularly check fixity and completeness of digital objects
	Manually assessing risk on content in a regular basis (some by using characterization tools and policy-based risk assessment)
	No content monitoring for risk assessment done
	Manually monitoring ingest material and process as it is provided
	No monitoring of ingest material and process
	Object retrieval failure warning on on-site access
	System web user interface access statistics
	No monitoring of access
Organization	Direct feedback from managers about changes in organization and internal discussion on regularly scheduled meetings
	Low level policy-information on particular activities
Knowledge Base	Manual monitoring of external registries and literature (e.g. PRONOM)
	Informal communication with colleagues of other organizations and contact with research community enabled by the participation in research projects
	No monitoring of any external knowledge bases
	Internal wiki with manually inserted information about collections

<sup>12</sup> The exception is system monitoring, for which some automatic monitoring tools are used. Monitoring storage and system operations on a bitstream level, however, is not considered within scope.

	Internal experimentation of tools
	No monitoring or register of any information in an internal knowledge base
User Community	Monitor user community directly by on-site user contact and feedback
	Direct consultation of user community because of close relationship
	Monitor user initiated email feedback
	No monitoring of user community
Producer	Manual monitoring of producers
	Producer direct communication because of close relationship
	Institution is its own and only producer
Technological Environment	Manual monitoring of producer technology, international technological trends and national or international agreements
	Having an active part of the development of new technology specifically related to digital preservation or format standardization (e.g. PREMIS, NeXus)
	Ad-hoc monitoring of technologies and best practices by the IT department

### 1.3 Goals

The high-level goals for the Watch component are provided below. They will direct the Watch component development.

1. *Enable the planning component to automatically monitor entities and properties of interest.*  
Based on an output plan, the planning component will define a number of questions and notification events. Regarding the notifications received from the Watch component it may change the plan and eventually pose additional questions.
2. *Enable human users and software components to pose questions about entities and properties of interest*  
Components and human users will be able to pose questions to the Watch component and receive a notification when there is an important change.
3. *Collect information from different sources through adaptors*  
Different sources will be relevant for the Watch component. Each source of information will provide specific knowledge to the Watch component, which will be able to collect this knowledge through adaptors.
4. *Act as a central place for collecting relevant knowledge that could be used to preserve an object or a collection*  
Information for object/collection preservation will be collected by the Watch component. Users can access the Watch component in order to gather information about a variety of aspects.
5. *Enable human users to add specific knowledge*  
Human users will be one of the sources of information and therefore specific interface will enable them to add specific knowledge to the Watch component.

6. *Notify interested agents when an important event occurs*

Besides enabling users to query information, the Watch component will enable agents to set up triggers of when they want to be notified. In that way users do not have to constantly check if there is a change in a certain property.

7. *Act as an extensible component*

The Watch component will provide easy way of supporting new sources of information and providing new possible questions that can be posed to the system.

## 1.4 Concepts and terminology

The term **Watch component** designates the whole watch system with all its services, modules and interfaces. To understand the component and its context of use, we define the **Planner** as its primary client to be either a planning component or a human actor.

To express interest about some specific information, the Planner will pose **Watch Requests**. These will consist of one or more **Questions** about specific properties that are each related to a specific Entity. An **Entity** is an instance of predefined types also referred to as **Entity Types**. Each **Property** will describe a specific aspect of the Entity and determine the possible range of **Property Values**, i.e. measurements in time.

The Planner can specify how the changes of Property Values should be handled by installing Triggers. **Triggers** contain conditions about the measured Property Values and notifications that are sent when some specific (or significant) changes occur. The decision whether a change in a Property Value is significant or not is made by an **Internal Assessment**, which evaluates all conditions of each Trigger in a Watch Request. Optionally, the Internal Assessment might rely on an **External Assessment** – a service provided by an external software agent such as the planning component. If the assessment shows that a change event is significant, this denotes a **Significant Event**. In this case, the Triggers of the related Watch Request get fired, i.e. the Notification(s) are sent out to the Planner. A **Notification** is a way of contacting the Planner when a Significant Event occurs. An example Notification would be a simple email.

Essentially, the Watch component consists of four main parts – the Watch Central Service, (a number of) Source Adaptors, (a number of) Monitoring Services and an internal Knowledge Base. In order to pose Watch Requests, the Planner will use the **Watch Central Service** or **Watch Service** for short. This is a central reference point between the system and the clients. **Source Adaptors** are responsible for gathering information from different sources and converting it to the internal data model of the Watch component. A **Monitoring Service** collects data from the Source Adaptors and pre-processes it if necessary. This means the data can be aggregated, relations to other pieces of information can be made, etc. The information is stored into an internal **Knowledge Base**, which is accessed for answering the Questions asked by the Planner(s).

Table 2 summarizes these concepts and provides information about further notions that are related to the Watch component and are used throughout the document.



Table 2 - Summary of concepts relevant for the Watch component

Concept	Description
Planner	An agent with an interest in the change of the state of the world over time. This can be either the planning component or a user client.
Watch component	Denotes the whole Watch System, with all its sub-components, parts and elements.
Watch Source (or Source)	A Source represents certain aspects of the world for which there exists a known way of investigating certain properties that represent these aspects. Sources can be internal or external, i.e. they can be part of the organization responsible for preservation or part of the outside world.
Source Adaptor	A software unit that is able to collect specific information from a specific source and has the capability to convert/adapt it into a representation that fits the internal data model of Watch.
Monitoring Service	Each Monitoring Service monitors a specific Source with the purpose of answering one or more specific Questions. For this purpose it uses one or more Source Adaptors. It might aggregate or relate different pieces of information.
Watch Service	The Watch Service is the central point of access to the Watch component. It holds a list of the Monitoring Services in the Watch component and a list of the posed Watch Requests. This service acts as a controller of the Watch component.
Knowledge Base	The (internal) Knowledge Base represents a sub-component that contains all knowledge that was retrieved from the sources and stored within the Watch component. This knowledge is used for evaluating the conditions and answering the questions.
Entity Type	An entity type describes the type of an instance. It groups instances of the same type and helps the Planner to pose meaningful Watch Requests. Some examples are: Format, Preservation Action, Experiment, etc.
Entity	An entity is a concrete instance of some entity type. E.g. 'ImageMagick v1.0' is a concrete instance (an entity) and has the entity type Action component.
Property	A Property defines a specific part of Entities in the world that have the same Entity Type, i.e. it represents some characteristic of these Entities. It can specify a data type for its measurements.
Property Value	Each Property Value is the atomic measurement of a property measured in a specific point in time.
Question	Questions are predefined points of interest related directly or indirectly to Sources and Properties. The Questions can be parameterized in order to offer some flexibility to the Planner.
Watch Request	A Watch Request is a composition of one or more Questions, created by a Planner.
Condition	Conditions are simple logical conditions that can be composed to build more complex ones. These are used during Internal Assessment of the answers and are always related

	to the Questions in a Watch Request.
Trigger	A Trigger is a unit that contains Conditions used during Assessment and Notifications that are sent when the Conditions are met.
Notification	A Notification describes what should happen when a Trigger is fired. Each Notification has a type, e.g. Email, Push Notification, etc.
Assessment	It comprises of Internal Assessment and the External Assessment. It denotes the act of evaluating the change events and deciding whether they significant or not.
Internal Assessment	It will use only the Condition(s) of the Trigger and internal knowledge to assess the situation and decide how to act.
External Assessment	It will use not only the Trigger Condition but also additional (global) knowledge that cannot be encoded into the Condition in order to assess the information. This module will <u>not</u> be part of the Watch component.
Change Event	Each time a new Property Value is provided for a Question that is different to the previous one, a Change Event is raised. This event has to be assessed in order to evaluate its significance.
Significant Event	A Change Event that is judged significant by Assessment and which thus requires the attention of the Planner.

Additionally, Table 3 provides a list of more generally applicable terms and definitions established in the PW sub-project of SCAPE.

Table 3 - General Terms and Definitions for Planning and Watch

Concept	Definition	Reference(s)
Preservation Planning	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.	(Antunes et al., 2011)
Repository	A system (of systems) that comprises ingest, access, content storage and preservation operations.	
Preservation Plan	A preservation plan defines a series of preservation actions to be taken by a responsible institution due to an identified risk for a given set of digital objects or records (called collection). The Preservation Plan takes into account the preservation policies, legal obligations, organisational and technical constraints, user requirements and preservation goals and describes the preservation context, the evaluated preservation strategies and the resulting decision for one strategy, including the reasoning for the decision. It also specifies a series of steps or actions (called preservation action plan) along with responsibilities and rules and conditions for execution on the collection. Provided that the actions and their deployment as well as	(Becker et al., 2009)

	the technical environment allow it, this action plan is an executable workflow definition.	
Preservation Action	<p>A preservation action is a concrete action (usually implemented by a software tool) performed on content in order to achieve preservation goals. For example, a migration of content to a different representation using a certain tool in a certain configuration and environment; the replacement of a viewer; the execution of a legacy viewer in an emulation environment... Preservation actions are thus the main object of interest for planning, which has to find the best action among a number of choices.</p> <p>A composite Preservation Action may consist of elementary Preservation Actions and may include conditional branches and other control-flow constructs (e.g. in an executable workflow)</p>	
Preservation Characterization	The process of measuring properties of content that are perceived of interest and documenting these measures in a structured form.	(Antunes et al., 2011) where this is called "Analysis"
Quality Assurance	The process of measuring properties of renderings/performances of content (by analysing the content, by simulating a rendering, or by other means...), comparing renderings and delivering measures of similarity according to specified criteria of interest in a structured form.	(Antunes et al., 2011)
Collection	In planning, a collection is the set (or aggregation) of objects that a plan is being created for, i.e. the assemblage of artefacts (objects, node sets of research assets, ...) aggregated on the basis of some common characteristic. (Common characteristic may be (in RDS) a common preservation function (enable you to perform a certain type of analysis; enable you to render a certain artefact;...) or (in object-based scenarios) common properties of the content to be preserved (format, features, ...)).	(Becker et al., 2009; International Council on Archives, 2000)
Collection Profile	A structured description of the properties of a Collection	
Content Profile	A structured description of the properties of a set of objects. This may include more than one Collection. The set of objects may be partitioned into Collections for planning purposes	
Criterion	A criterion is a relevant decision factor referring to a measurable property of a Preservation Action or its outcome, whose value can be determined independently of other criteria.	
Objective	An Objective is a quantified goal that is decidable. It can consist of multiple Criteria and a corresponding assessment. Given measures are known for each Criterion associated with an Objective, it must be possible to decide the degree of fulfilment of the Objective. Objectives can consist of a combination of other Objectives and Criteria.	

Driver	An external or internal condition that motivates the organization to define its goals.	(The Open Group, 2009)
Organization	A self-contained unit of resources with line management responsibility, goals, objectives, and measures	(The Open Group, 2009)
Preservation Policy	<p>Preservation Policies are governance statements (about constraints, goals, preferences, directives) that constrain or drive operational Preservation Planning, but may also have other effects outside of operational planning.</p> <p>In particular, we distinguish between high-level <i>guidance policies</i> which are neither enforceable nor quantified; <i>control policies</i> which are specific and can be represented in a semantic model to a degree sufficient for automated reasoning; and <i>procedure policies</i> which are practicable, but not necessarily represented in a semantic model. Procedure policies relate control policies to their originating guidance policies.</p> <p>Only the control policies are guaranteed to be represented in the policy model so that Watch can automatically monitor them.</p>	
Watch	The capability to monitor external and internal entities for changes having a potential impact on preservation and to provide notification.	Derived from (Antunes et al., 2011; CCSDS, 2002; Sierman & Wheatley, 2009)

## 2 Requirements

To gain more understanding of the functional requirements for the Watch component, this section describes significant events and triggers that might be of interest to Watch. Afterwards to extend this set, specific information sources with accompanying triggers and events are identified and described.

The section further presents a simple real world scenario that is used to identify potential questions for Watch and provide a perspective for the subsequent more detailed discussions.

### 2.1 Significant events and triggers

A number of *significant events* can be related to business and organizational drivers such as the influencers described in Table 4 (from the SHAMAN RA). However, not all such events lend themselves to a classification in *one* category, since they may relate to several drivers at once. For example, the emergence of new external knowledge about a migration tool that is in usage in a Content Repository can raise a risk that was not previously recognized in the organisation.

Table 4 - DP drivers as specified in the SHAMAN Reference Architecture

Internal	Business Vision	Goals, Scope of designated community, etc.
	Resources	Infrastructure (e.g., operational costs, expertise needed), Hardware (e.g., operational costs, technological capability), Software (e.g., operational costs, technological capability), Staff (e.g., expertise and qualifications, commitment)
	Data	Volume, Structure, Representation, Semantics, etc.
	Processes	Dependencies, Responsibilities, Alignment, etc.
External	Producers	Demand satisfactions, Content, Technology, Trust and reputation
	User community	Technology, Knowledge, Demand satisfaction, Trust and reputation
	Contracts	Deposit, Supplier and service, Interoperability, Access, etc.
	Supply	Technology, Services, People
	Competition	Overlap of: Services, Content, User community, Producers, Technology, Mandate, Rights, Funding, Capabilities
	Regulation and mandate	Regulation/Legal constraints, Embedding organization regulation, Mandate, Rights and ownership, Certification, Funding

Triggers can be very specific for the organizations/users using the Watch services. However, a generic model of the changes of the world has to be provided in order to anticipate all categories of triggers that are required.

Previous work in the Planets project has started such a classification. In the following, we build on this classification and align it with results of the preliminary User Group (UG) surveys and the Testbed scenarios in SCAPE. We further combine it with some of the drivers given above that have not been considered in the Planets analysis. As the coverage and precision of answers in the UG survey varied considerably, some generalizations and optimizations were made.

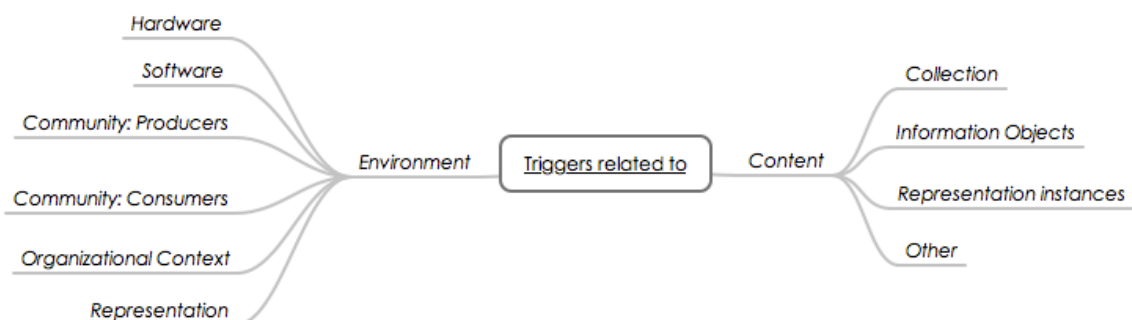


Figure 4 - Categorization of triggers

Table 5 shows a principal categorization of triggers derived from the original work done in Planets. Essentially, key events can be arising from the content or any aspects of its environment (as depicted in Figure 4). Content-wise, we can distinguish between events arising from content collections (such as new ingest activities creating new objects), events about specific information objects (such as the

failure to render certain objects), events about the representations of objects (such as validity) and other aspects.

Environment-related events encompass the entire organization and its systems, hardware, software, processes, policies and people. It furthermore includes all external influencers such as those specified in Table 4.

In Table 5, we list examples of content related triggers (each row) with their drivers, events and example causes.

As an illustration, consider the trigger *Information Objects: changed valuation*. The estimated value of specific content may be changed by collection management at any point in time. If a policy specifies certain preservation levels or risk thresholds depending on the value of objects, this will be reflected in plans and may need to be adapted, as objects may now be more or less valuable and require different treatment. The valuation of content will be reflected in the SCAPE policy model.

A different example is presented by Representation validity. Consider an Ingest process that checks format conformance according to a profile specified in the policy model. If this policy-based validation fails, this is an indicator that the submitted content is invalid. It will be detected during the Ingest process, so the source that needs to be monitored is the Ingest process. The root cause, of course, is not in the Ingest process, but likely in the Producer's realm.

Table 5 - Exemplary content-related Triggers

Driver	Event	Example Cause
Collection	content corrupted	bit rot/failed disks
	new content	new digitization and ingest activity
	new or changed information about content	new characterization activity
	growth/size changes unexpectedly	producer increases production, ingest process increases efficiency...
Information Objects	object(s) with new properties	producers technology changes
	changed valuation	changes in collection management may change the valuation of certain parts of the repository's content
Representation	syntactic or semantic interpretation fails	obsolete format, protected object, lost keys...
		preservation network fails
	representation validity	failed migration process, corrupted content, submitted content is invalid
	changed representation assessment	Format support is officially terminated

In an analogous way, Table 6 describes triggers relating to the environment and provides examples for causal events. To illustrate this again with an example, consider the row *new software*. Quite regularly, desirable migration actions are postponed because the available software components do

not provide sufficient quality or sufficient control over operations to guarantee authentic content upon migration. New versions of preservation action components often offer improvements over previous versions. They can potentially present a much better solution, and there is thus interest in being notified about their publication. Similarly, it is of high interest when an automated tool becomes available that is able to perform analysis or Quality Assurance tasks for properties that are of relevance in a certain scenario. If these properties previously had to be evaluated manually, such a tool then offers a new possibility to conduct automated planning experiments.

The indicator for such events is when new (versions of) software fitting certain criteria are added to the component catalogue. This can be detected by monitoring the component catalogue, which represents a Watch Source.

A less typical example is presented by Hardware costs. Typically, storage costs are quantified in the yearly costs per storage amount, such as per GB. These costs are an important decision factor for planning and need to be monitored over time. If storage costs change, this will not be directly monitored. Instead, the storage costs will be reflected as factual statements in the policy model and can thus be monitored as part of this policy model. Any cost model can be used as an input to update the policy. If the storage costs rise, an event can be generated that leads to a reevaluation of plans and an assessment of adaptations that may be required for offsetting the increased costs. The indicator here may be a change in the factual statements of the policy model.

Table 6 - Exemplary environment-related triggers

Driver	Event	Example Cause
hardware	storage warning	full disk
	hardware failure	bit-rot occurred
		peripherals stop working (e.g. external controllers for the video game collection)
	capabilities change	optimized internal infrastructure leads to new technical opportunities
iPads are bought by management for the reading room		
costs	increased storage costs may need to be balanced by adapting preservation plans	
software	new software	new migration software for specific formats used in the repository new analysis/characterization software for a certain content type / format new QA software for a certain content type / new monitoring service for a question of interest new software version release
		new testing results for a (action/analysis/QA/monitoring) software used in the Content Repository
	new knowledge about software	change in software dependencies
	new repository system/version	new software release, acquisition of a new system
capabilities	optimized internal infrastructure leads to new technical opportunities (e.g. faster throughput in organizational SOA)	

	costs	reduced costs of running software lead to new opportunities for improving cost efficiency of preservation operations
Community: Producer	Change in community scope	new producers added with a different background
	Change in demand	producers want change in the way they deposit material
	Change in content	producers send different material
	Change in technology	producers use different technology to create material
Community: Consumer	Change in community scope	new consumers added with different background
	Change in demand	consumers want change in the way they access material
	Change in knowledge	consumer community knowledge base has changed
	Change in technology	consumer community's preferred viewer environment has changed
Organizational context	policy change (goals, strategy, constraints...)	management change/regulatory change
	legal change	extension of mandate caused by legal deposit act
	regulatory change	standard gets revised
		new standard gets approved
		new regulation has to be applied
	supplier change	service supplier goes bankrupt
	contract change	supplier contract runs out and is not renewed contract with service supplier has to be valorised and raises above cost threshold
	other organizations	Partner organization changes policy in a way that affects risk assessment on organization's holding
Competitor (with overlap in content and/or community) introduces cutting edge access features		
representation	new format (version)	New format for a content type is developed
	new/changed knowledge about format	format gets standardized
		new risk detected

Note that the goal of Watch in SCAPE is to create *automated* mechanisms. We thus do not foresee significant effort to be invested in managing manual changes. However, some mechanisms may be developed for managing such changes in aspects where they can substantially improve decision-making efficiency. This may in particular apply to the policy model.

## 2.2 A planning-and-watch scenario

As an initial starting point for monitoring, we will consider the case of a preservation plan. At the point of decision-making, a choice is made for a certain preferred action to take, based on goals and



requirements of an organization, evidence and its assessment, and the choice of possible actions. In the future, any of these key factors may be subject to change:

- The goals and requirements of the organization may shift,
- The available evidence may grow and reveal additional insights and deviations from the evidence available at the time of the original decision,
- The assessment of certain qualities encountered in the evidence base may change, subject to additional insights and knowledge gathered in or outside of the organisation, and
- The choice of possible actions, or supporting actions (such as automated Quality Assurance) may expand or be reduced.

Furthermore, the operational deployment of the chosen action may, for whatever reasons, not be operating in a fully compliant way, i.e. deviating from the evidence gathered during decision-making.

Hence, the (rather ambitious) goal of an ideal monitoring system would be to **monitor compliance of operations deployment; risks of the scenario that concern the requirements and the deployment of the specified actions; and opportunities related to the potential choice of actions.**

We will in this section take a well-defined simple planning scenario as a starting point and derive examples for each of these categories in order to illustrate the basic requirements for the Automated Watch component. This scenario provides an understanding of the needs for the Watch component in terms of possible questions. However, it will not be considered as presenting an exhaustive list of all possible Watch Questions.

### **2.2.1 The British Library Newspapers plan**

The British Library has 2 million images of scanned articles dating from the 19<sup>th</sup> century onwards. Initially the entire collection was in the format TIFF-5. A Preservation plan was created for two reasons; first, to discover if there is a better alternative for providing a long-term access to the collection, and second, to see if it is possible to reduce total ownership costs with minimal preservation risks.

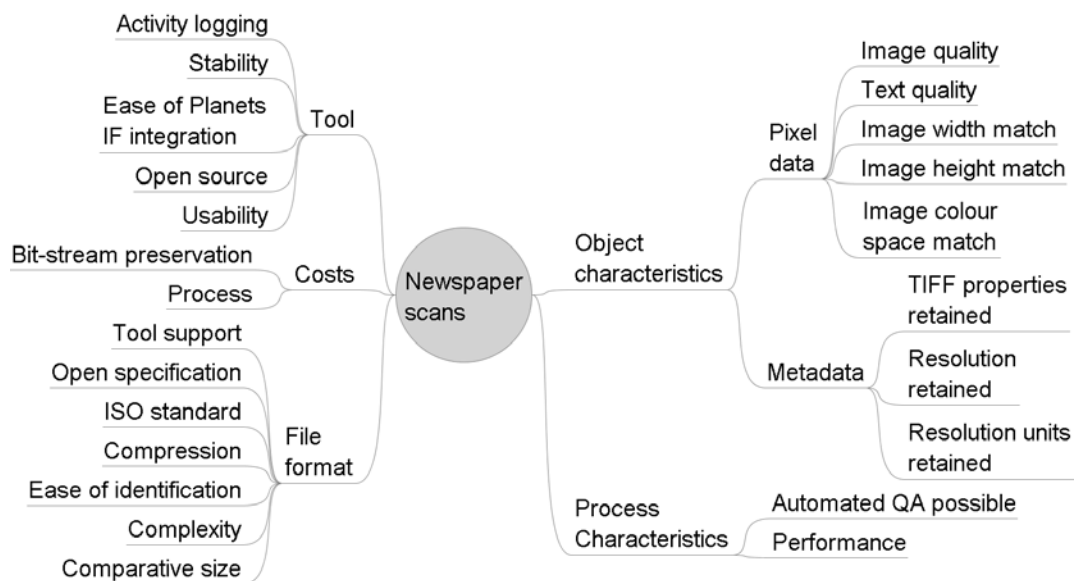


Figure 5 - BL Newspaper objective tree

Actions initially considered in the plan included the migration to a number of image formats (TIFF-6, PNG, BMP, GIF, JPEG, uncompressed JPEG2000, and JP2 lossy compression (80, 90 and 95)) using the ImageMagick tool. GIF and BMP were quickly discarded as part of the planning workflow because of the palette size and the file size. All other actions were evaluated according to the measurable criteria shown in the BL Newspaper plan objective tree (Figure 5). Actions were evaluated in a controlled experimental setting, and information from external sources was collected. In the objective tree this is clearly separated. Object Characteristics and Process Characteristics criteria are gathered from experiments, while Tool, Costs, and File Format criteria are fetched from external sources of information. For each criterion, the utility function maps each possible measure to a score. Summing up all scores, a final recommendation is provided by the Planner. The following discussion is derived from the above and a number of similar plans, all of which are discussed in (Becker & Rauber, 2011).

Assume that the decision is to *“migrate from TIFF v5 to JPEG2000 with ImageMagick v6.1 (config settings: compressionType=0, compressionQuality=100)”*.

During the time when the plan is active, changes may arise which could affect the assessment of options in the plan. For example, tool support for JPEG2000 will likely increase. The Planner would like to know when certain requirements are met. There are numerous other examples that could change. At the same time, it is not feasible for a human user to manually monitor all of these factors. Furthermore, when the plan is executed (whether upon ingest of an object or on a whole collection at once) the Planner will be interested in **monitoring operational compliance**: The planning decision was taken based on controlled experiments on sample content. These experiments will have shown that the input and output of the migration as well as the process behaviour are fulfilling the requirements and constraints of the organisation. However, they have been performed on a limited subset of the whole collection and do not provide 100% assurance that something may not go wrong when executed in the real environment on the full set of content. Therefore, the Planner will request that compliance is monitored upon plan execution and request the Watch component to notify him when there is a significant deviation from expected levels of service. To achieve this, a number of Watch Requests will be defined.

The objective tree such as the one in Figure 5 provides the starting place for deriving Watch Questions. The conditions that will define when there is a need for a notification are derived from a utility function for each criterion. Object Characteristics and Process Characteristics are focusing on experiments and operations as a source of information. Tool, Costs and File Format criteria will need external sources. Note that the specification of these conditions takes part within the planning procedure as an essential step in defining the preservation plan (Becker et al., 2009). The Watch component itself does not need to know anything about utility functions.

Since each action is evaluated against this objective tree (Figure 5), corresponding questions can be defined to monitor **compliance** of and **risks** affecting the chosen action and **opportunities** of alternative actions. It is foreseen that the majority of monitoring conditions will be derived automatically in the planning component from the decision criteria specified in the objective tree (Hamm & Becker, 2011). However, the Planner can optimize these, in particular to reduce the amount of Watch requests posed.

A Planner might have decided not to migrate to JPEG2000 because the tool support at the moment is limited. Therefore, he will pose a Watch Request such as “Notify me when the tool support of the JPEG2000 format changes to *widespread*”. This event would open an opportunity for action. Similarly, the current tool support of TIFF version 5 (which is the format of the whole newspaper collection) is widespread, but a change of this would expose the Planner to a risk, and he would again be interested to get a notification. Thus, he may pose a Watch Request “*Notify me when the tool support of the TIFF format changes to limited or none*”. Changes related to other formats could be monitored as well, since these can be considered as opportunities.

The following tables are examples of possible monitoring conditions derived from decision criteria. Note that some of these criteria have been slightly rephrased in the course of (Hamm & Becker, 2011) and thus may deviate slightly from the original criteria mentioned in the plans. Table 7 shows an example list of format measurements to be monitored. If the specific condition is met, a Change Event is raised.

Table 7 - Examples of format measurements and possible conditions for the chosen target format

Measurement	Condition
Tool Support	number of tools that are known to read this format drops below 5
Open specification	openSpecification=NO
ISO Standard	Format ceases to be international standard
Complexity	complexity= MEDIUM or complexity=HIGH

Table 8 lists examples of monitoring conditions related to tools, processes and representation instances.

Table 8 - Examples of tool, process and representation instance measurements and possible conditions

Measurement	Condition
Activity Logging	number of characters logged by operational process is <10 for any single file
Stability	tool crashes during execution
OpenSorce	openSource=NO
Usability	usability=EXPERT

Comparative filesize	relative filesize of migrated copy > 135% of the original file
Validity	Migrated object cannot be validated for format validity by JHove
Object not well-formed	Migrated object is not considered well-formed by JHove
Performance	Accumulated average processing time per MB > 2 sec OR Processing time for one object > 20 sec

New file formats that could be used to represent the content at question can be considered as opportunities. Therefore a Planner may pose Watch Requests about new file formats that he can use: *“Notify me when there is a new image format”*. However, the Planner will likely be not so much interested in new file formats in general as in new tools that can do a specific migration. Even more interesting will be evidence pointing to the applicability and quality of certain preservation actions with regard to the content at question. Hence, the request *“Notify me when there is a new tool that can accept as an input format X and output format Y”* is valid and can be combined with more specific filters that ask for experiments conducted on such tools. To avoid excessive notifications without direct value, the Planner can ideally pose requests about experiments provided by others: *“Notify me when there is a new experiment on image conversion from TIFF-5 to JPEG 2000 where at least 100 objects were converted successfully and validated for image width, bit depth and pixelwise image identity”*. Results from experiments can be made available to other clients by storing them in a results database. This enables automatic knowledge exchange between content holders, which can be very valuable to all parties.

A valuable monitoring feature requested by organisations is to enable the collection and exchange of statistics about collections in order to ask questions such as *“How many collections exist with properties X, Y and Z?”* or *“Notify me when the number of organisations holding format X drops under 5”*.

Finally, a different group of opportunities present themselves when we consider the degree of automation in the realms of characterisation and quality assurance. To date and for the foreseeable future, planning decisions will be partly based on criteria that can only be evaluated manually. However, increasing automation in SCAPE and beyond will make available technical means to automatically measure certain properties of interest, such as format validity, JPEG 2000 profile compliance, or the layout similarity of page-based text documents, using characterisation and quality assurance components. The availability of such components could thus be monitored to notify interested parties when certain automation features become available. Similarly, experiments that test the precision that certain components exhibit when applied to content with certain characteristics can be of interest to responsible Planners.

A summary of conditions is given in Table 9. We distinguish types of criteria according to their sources of measurement, as discussed in (Hamm & Becker, 2011). The table shows examples for monitoring the compliance of the chosen action, risks and opportunities affecting actions, and risks and opportunities for characterisation and Quality Assurance.

Table 9 - Monitoring compliance, risks and opportunities: A summary of conditions

Type of criteria	Compliance of chosen action	Risks and Opportunities of actions	Risks and opportunities of characterisation and QA
Transformation Information Properties	Image not pixelwise identical	Exists action with pixelwise identical images for TIFF-5,	Exists new QA component for measuring image

	Image width not equal	JPEG2000	equality  Exists new experiment showing false measures for QA tool used in the planning experiment when comparing TIFF-5 and TIFF-6
Representation Instance Properties	Object not valid  Object not well-formed  Comparative filesize > 1.8 OR accumulated average comparative filesize > 1.35	Exists action producing valid JPEG2000 object from TIFF-5  Exists action producing well-formed JPEG2000 object from TIFF-5  Exists action producing JPEG2000 object from TIFF-5 with filesize <= 1.35	Exists new PC tool for checking validity  Exists new PC tool for checking if objects are well-formed
Action Runtime properties	Processing time per MB > 5 sec	Exists action producing JPEG2000 object from TIFF-5 with speed >25% better than chosen action	
Format properties	Number of readers supporting target format drops below 5	Number of browsers supporting JPEG2000 rises above 2  Exists experiment with chosen action in different configuration with higher performance	
Other action properties	Stability of chosen action is reported to be very low in large-scale experiment migrating TIFF-5 images	Exists action producing JPEG2000 object from TIFF-5 with higher stability rating than chosen action  Exists experiment with chosen action in different configuration with higher stability rating	

It is likely that Watch Requests Can be grouped to increase efficiency and cohesion. For example, the entries regarding alternative actions fulfilling the organisation's needs in the opportunities column effectively could be compiled into one Watch Request.

### **2.3 Policy-driven Watch**

As a different starting point, consider a case where no planning has taken place. The organisation runs a repository and has just defined policies. The two facts alone should trigger a monitoring activity. For example, a policy may state that for every set of objects belonging to one format, a plan needs to be defined if there are more than N objects in that set; or that objects that are in non-standardized formats shall be normalised to certain formats (pre-specified or not), unless that would lead to a loss of content.

These policies have to be monitored in relation to the collection profile in order to raise alerts. An alert would thus trigger a planning activity that defines the appropriate response to an identified issue. On the other hand, the identification of conditions and triggers can be very scenario-specific and more difficult to generalise. Furthermore, the monitoring has to not only collect information from one source, but also relate in a more sophisticated way the information of several sources (such as the policy model, the repository, the collection profile, and the format registry) to each other. The feasibility of this has to be investigated as soon as the first iteration of the policy catalogue is delivered.

### **2.4 Sources of information**

As it is seen in the Section 2.2 (A Planning-and-Watch scenario) the Watch component will consult number of external sources to collect knowledge. This section gives an overview of these sources in terms of information they provide.

Up until now several possible source types have been identified:

1. format registry
2. component catalogue
3. policy catalogue
4. repository
5. content profile
6. experiment results
7. web browser snapshots
8. simulator
9. human knowledge

Maybe there are some unforeseen source types or there are going to be in future. This will not pose any kind of problem to the Watch component because it will be designed to allow the addition of new sources.

### **2.4.1 Format registry and component catalogue**

Format registries and component catalogues are imagined as places where relevant information about formats or components is stored. Each format (component) is represented as an entity with properties that describe that entity.

Watch requests posed about formats or components will be usually about these format or component properties. Simple examples are given in BL Newspaper plan scenario, described in Section 2.2.1. There it was shown that the Planner is interested to fetch any significant change from the format registries that could be characterized as a risk or an opportunity for his plan. Furthermore the Planner is also interested in new formats or components that could be used. Even experiments that provide proof about usefulness of these new formats and components are of relevance here. Hence, it is important to provide linkage between format registries, component catalogues and experiment results.

There are two different things that Watch component will observe from format registries and component catalogues.

1. changes in properties of existing formats and components
2. appearance of new formats and components and linking them with experiments

It should be noted that questions of interest for the component catalogue might not only be related to migration or emulation components, but also to components that support these. For example, new QA tools that are able to measure significant properties hitherto not covered are of high interest, since they can increase automation and scalability.

Examples of this type of source are the PRONOM and P2 registries. However, the crucial point is the coverage of information, and current information sources are still severely lacking in this regard.

### **2.4.2 Policy model/Policy Catalogue**

Changes in the policy model can often have an impact on the preservation plans related, so it is important to keep track of policy changes. Here only one type of watch request will be relevant, such that a user is notified when there is a change in the policy model that has an effect on plans.

Policy models may not change often, so this source of information might work as a push source where the policy model will notify Watch component about changes.

The policy model in SCAPE is currently being designed. However, it is clear that the policy model will focus on aspects that are well-understood and specific, such as format assessment rules. Hence, it should be possible to apply simple mechanisms for automated monitoring of compliance of the status of planning and operations to specified policies. This will be analysed in greater detail as soon as the first iteration of the policy model becomes available.

### **2.4.3 Repository**

Given the repository itself is an operative system, monitoring is limited to the information actually provided by that system. Hence, in order to be able to support that monitoring, the repository requires the capacity to provide operational statistics upon request.

Questions can correspondingly be posed concerning four different aspects:

- Repository properties (capacity, used/free space, etc.)

- Collection status (e.g. number of elements in a collection; is there a defined plan for a certain collection; growth rate of a defined set of objects)
- Objects and their properties (e.g. number of objects corrupted, specific properties about objects, etc.)
- Operations (e.g. number of corrupted elements encountered in ingest, average access time using migration upon access, number of failed access requests...)

#### **2.4.4 Experiments and results**

During the planning phase, experiments are executed on a subset of the whole collection to provide insight into potential alternative actions. These experiments provide valuable knowledge not only for the current plan but also for any future usage. They are executed only on a subset of a whole collection, but processing this subset can still take significant amounts of time. Moreover, the experiment results will often be validated and amended manually and are therefore particularly valuable. Therefore, the Planner is going to have the possibility to publish his experiments so that the results can be accessed. If this is widely done, it can provide a significant amount of knowledge.

Additional content concerns the results of plan execution. The first aspect that is to be monitored here is operational compliance. It is especially interesting to see when experiments suggested that operations be compliant but results show otherwise. Other Planners can use such information to detect possible risks in own plan. Furthermore, these results can be used to monitor community trends as well. If the Planner notices that a lot of content holders are migrating from a format occurring in his collections to another format, he could see this as possible risk or opportunity for his content as well.

Finally, any other type of experiment conducted in the SCAPE Platform is potentially of tremendous interest. This includes experiments analysing the performance and accuracy of characterisation and QA components as well as action components, even if these experiments are not primarily conducted in order to address a planning scenario.

Results of experiments are potentially the largest and fastest growing information source and probably the source that requires most sophisticated filtering. Furthermore, to exploit all possibilities, links between information from experiments, the format registry and component catalogues need to be provided.

#### **2.4.5 Content profile**

The process of content profiling consists of three steps -- characterization, aggregation and analysis. Repositories provide the meta-data and some statistics such as the count of objects or formats of the content and the collections they keep, however these are not always sufficient in order to conduct planning efficiently. Furthermore, due to the volumes of the data, content holders are often unaware of the type of content that they have, e.g. content in web-archives.

A content or collection profile provides statistical data about digital content of any type. A profile offers an aggregated view of the collection metadata and provides the basis for detailed analysis over the content. For example it offers distributions of characteristics in the collection and combined with other tools, it can provide valuable information such as trends and the search for a representative subset of the collection based on a specific set of predefined properties. The quality of the analysis depends on the characteristics that can be extracted from the different tools, but even an



aggregation or a distribution over a few characteristics can provide meaningful information about the content.

An example question might be: what is the change in the distribution of a specific property (e.g. format) in a collection over time?

A content profile tool could enable content holders to submit/extract information about their collections. If enough content holders provide simple statics and aggregation of their content, such a source can potentially have a large value for them. On the one hand, they can monitor the formats other content holders use and potentially find out about some risks and opportunities, but on the other hand it would enable a worldwide format watch as well as a number of other analyses that could be of high value.

#### **2.4.6 Web browser snapshots**

To apply methods developed as part of the Automated Watch in SCAPE to Web Archives and extract information from this heterogeneous content, SCAPE partners are developing a Web Content Watch adaptor service to automatically detect rendering issues and monitor file format obsolescence as well as emerging technologies.

To monitor Web Content, this service will rely on a snapshot tool that will allow the creation of a set of “reference web pages”. Snapshots of Web pages will be taken automatically from the live versions as closely as possible to harvest time.

Image analysis, QA and characterization tools will then allow automatic comparison of web pages to detect rendering issues and extract and store any useful information, about file formats issues in relation to browser versions for example.

This service therefore comprises action tools for rendering web sites; characterisation tools to characterise renderings; and QA tools to compare renderings. This snapshot service will, periodically or upon request, provide detailed reports about the comparison results regarding configurable inputs and browser platforms and specific measures and properties, which are yet to be determined in detail.

Statistics and preservation related information would be fed into the Watch component to improve detection of preservation issues and initiate preservation actions.

The service will be deployed and integrated with the Watch component either through a specialised adaptor or simply by depositing the results of the snapshot service in an experiments database.

#### **2.4.7 Simulator**

The simulator is a component that will be used to simulate the effects of a planning and watch. This is especially important for large-scale repositories where understanding of storage and computational resources is crucial. Integration of the simulator as a Watch source was not originally planned in SCAPE, but will be desirable if time permits. To detect potential risks and evaluate potential actions, the simulator can then be used as a source of information. Watch requests would be posed to detect in advance significant events for the repository. Two possible watch questions arise here:

1. Questions about a certain property in the future.
2. Questions about the potential outcome of certain preservation actions.

### 2.4.8 Human knowledge

Human users should be able to insert information about every possible entity (format, tool, experiment, repository object status, etc.).

## 2.5 Significant Events and Questions

In Table 10 we present some example questions and their conditions aligned with the triggers from above. In general we provide an Indicator that designates the event and a source by which it can be retrieved.

As an illustration, consider the trigger *Information Objects: changed valuation*. The estimated value of specific content may be changed by collection management at any point in time. If a policy specifies certain preservation levels or risk thresholds depending on the value of objects, this will be reflected in plans and may need to be adapted, as objects may now be more or less valuable and require different treatment. The valuation of content will be reflected in the SCAPE policy model. The Indicator thus is a change in the policy model. The Watch Source is the policy model itself.

For another example, consider the row *new software*. New versions of preservation action components can potentially present an improvement, and it is of similar interest when an automated tool becomes available that is able to perform analysis or Quality Assurance tasks for properties that are of relevance in a certain scenario. The indicator for such events is when new (versions of) software fitting certain criteria is added to the component catalogue. This can be detected by monitoring the component catalogue, which presents Watch Source.

Table 10 - Example questions and conditions

Event	Question	Indicator	Condition	Source
Content corrupted	Is there corrupted content in a collection?	Completeness validation fails	Log contains validation failure data	Repository
		Access fails	Access failure event reported	Repository User
New content	Is any content being ingested into the repository?	Ingest activity notices new content	The format of ingested content is different from the content profile	Repository Ingest Collection profiler
Growth/Size changes unexpectedly	Is the content size growing unexpectedly?	Rate of growth changes drastically in ingest	Growth of collection X exceeds threshold	Repository Ingest, Collection profiler
Size of collection grows	Do we have plans defined for all collections?	Mismatch between content profile, policy model, and set of plans	exists collection with <code>collection_size &gt; policymodel-threshold</code> and no plan defined	Repository, Policy model
Change in community scope	Are there any new or different producers?	New producer uses ingest process, new producers send new material	Exists new producer	Repository
Change in community	Are there any new or changed consumers?	New consumers use access	Exists new consumer	Repository

scope		process		
Changed valuation	What is the current valuation of collection X?	Change in policy model	Valuation changes	Policy Model
New knowledge about software	What is the current status of information concerning experiments about migration from TIFF to JPEG2000	Evaluation data in an experiment platform	Exists new experiment with specified entities and properties	Experiment results
New software	Is there any software for converting TIFF to JPEG2000 that is applicable to our server environment?	New software fitting certain criteria is tested within the experiment database	average(memory_consumption)>10	Experiment results, Component catalogue
New content	What is the most widely used text format within collection X?	New acquisition activity, new ingest activity	Mode (format) changes	Collection profile, Repository
Impending issue with storage size	According to the current repository status, what will be its size in the future time X?	Simulator prediction	In time X, the total size will be above a certain threshold	Simulator
Impending issue with storage costs	According to the current repository status, what will be its storage costs in the future time X?	Simulator prediction	In time X, the total costs will be above a certain threshold	Simulator
Simulator creates new prediction	What is the predicted lifespan of format X?	Simulator prediction	The obsolescence time is below threshold	Simulator
New format risk	What is the risk status for the format X	New risk defined in the policy model	There is a new control policy about required format properties	policy model

The list presented in Table 10 may not be complete and thus presents only a subset of the possible questions and sources. The intention is to give an overview of the areas of interest of the Watch component. On the other hand, it is questionable whether a full coverage of all these aspects within the timeline of the project is feasible, since some of these aspects are highly complex. The first step in development is to assess the value, feasibility, and required schedule for each of the different aspects to optimise resource usage and maximise benefits.

### 3 Architecture

This section revises these and provides an overview of the main sub-modules, their interfaces and UML diagrams where necessary. Note, however, that parts of the interfaces are still subject to change and that only high-level interfaces are defined here.

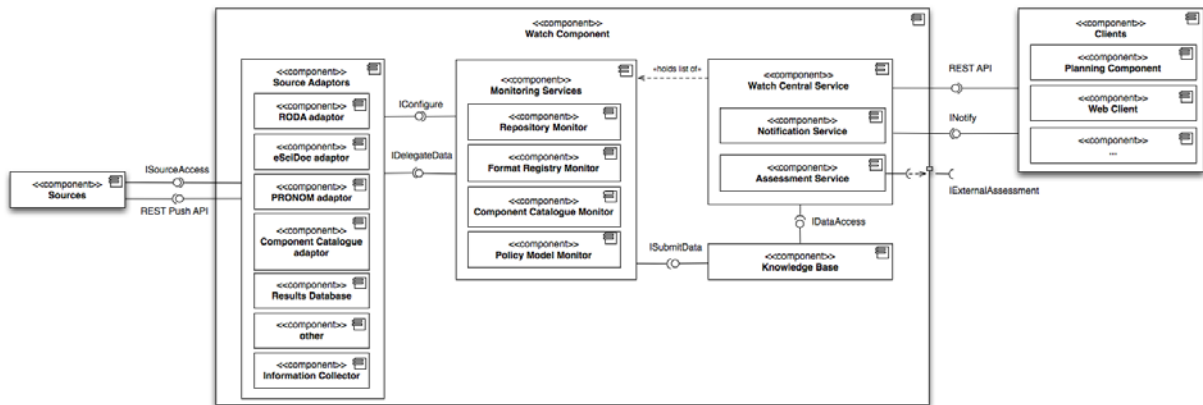


Figure 6 - High-level architecture of the sub components of Watch

In the following we describe an iteration of the data flow and its transformations looking at the big picture of the Watch component, by providing a real-world example. Figure 6 illustrates the high-level architecture of the Watch component and its sub-components as well as clients and sources of information.

In order to query and make use of the information managed by the Watch component, the Planner has one tool at hand – a Watch Request. Technically, there are two types of such requests, Synchronous (Sync) Watch Requests and Asynchronous (Async) Watch Requests. The former are used to query a specific measurement in a point in time (the current by default). They enable the Planner to make use of information in the Knowledge Base and block the client until the response is returned. The latter are used to monitor changes in these measurements, by specifying Conditions, which may trigger the execution of a Notification. As soon as they are submitted, the client continues its work and can rely that a Notification will be generated as soon as a change is detected – thus the name Asynchronous. These two notions will be defined in more detail later in this section.

Consider again the BL scenario and a Planner (e.g. the planning component) that is interested, whether the currently chosen preservation action described in a plan is still the best alternative. The Planner submits an Asynchronous Watch Request, which monitors specific properties for a new alternative, i.e. a new opportunity. Assume that based on the current knowledge the best alternative is to do nothing and keep the status quo. However, if a new format is found that meets specific criteria (e.g. its tool-support is widespread, the format is ISO-standardized and the compression is lossless), or a known format has changes in these specific criteria (e.g. becomes ISO-standardized) the Planner wants to be notified about such a change in order to re-evaluate the alternatives. In this example a simple email to a specified recipient will suffice.

In order to submit such an asynchronous Watch Request, the Planner uses the REST API (or the Web Client) and contacts the Watch Central Service. Essentially, the API call should contain the properties of interest, and the Triggers (Conditions and Notifications). As soon as this is done, the Planner proceeds with other tasks and can rely that he will be notified according to the specified Trigger.

If one of the formats considered in one of the alternatives of the plan, indeed becomes ISO standardized then this alternative would become the one with the highest score. Assume there is a source, monitored by the Watch component that contains the information about formats and their

standardization. The general idea is that such sources act as content providers to the Watch component and generate data in their own representation.

The corresponding source adaptor fetches the information about the standardization change via the `ISourceAccess` (Pull) API. Note, however, that not all sources of information will provide data via a pull model. In some cases it makes more sense to use a push model (REST Push API) and let the source push the information to the Source Adaptor (e.g. the Information Collector). In general, sources with large volumes of data will have to aggregate the data and use push whereas pull will be used on sources that generate small volumes of data (more frequently).

As soon as the Source Adaptor collects the new information the data is converted (if necessary) in order to fit into the internal data model and Knowledge Base as discussed in section 3.1.1. The `IDelegateData` interface is used to pass the data for further processing to the corresponding Monitoring Service. In this example this is the Format Registry Monitor.

In order to control and fine tune adaptors, e.g. the frequency with which they provide the data in case of a pull, each adaptor implements a special `IConfigure` interface that is used by the Monitoring Services.

Once the data reaches these, it might be pre-processed. For example it might be connected (related) to other chunks of information. At that point the converted and pre-processed data is submitted to the Knowledge Base and there it is persisted.

The Knowledge Base is responsible for data management and provides interfaces for storing and retrieving data. Furthermore, it might also provide different mechanisms for notifying other subcomponents when new data gets inserted or modified.

After the data is stored, the Watch Central Service is involved into the process again. It has to determine all Watch Requests that need re-evaluation based on the new knowledge. This could be done in two ways. Either there is a scheduler that decides which Watch Requests have to be re-evaluated or the Watch Service filters the Watch Requests based on the new knowledge provided. Both cases have their advantages and disadvantages and have to be discussed.

Assume that such a scheduler decides the Watch Request from the example has to be evaluated and passed to the Assessment Service for internal assessment.

The internal assessment evaluates the conditions based on the current knowledge and if they are violated a notification is created. If the conditions are met, then the process stops and waits for a new change. Some more complex requests might also need an external assessment module that is used via the `IExternalAssessment` interface.

Back in the BL scenario example, the internal assessment is sufficient and as the Condition is met (a format of interest is ISO-standardized) the Watch Service fires the triggers. This means it uses the Notification Service to generate and transmit each Notification in the Planners most preferred way, here to send the email specified at the beginning as part of the Watch Request.

After this is done the Watch Request is still active, and thus future changes will result in new notifications. If the Planner is not interested in this request anymore, then the Web Client or the REST API is used again, in order to remove or disable the request.

The next subsections provide more detailed overview of the different layers of the architecture and present a complete domain model, as well as high-level designs of the different sub-components and a list of front-end features and scenarios

### 3.1 Data Layer

The domain model can be divided in two parts: the part that models the information of the world which we will call **Knowledge Base**, and the part that models the questions about the world and as well as triggers (conditions and associated actions) which we will call **Watch Requests**. We will explain both parts separately and then join the models together in a complete view of the watch domain model.

#### 3.1.1 Knowledge Base

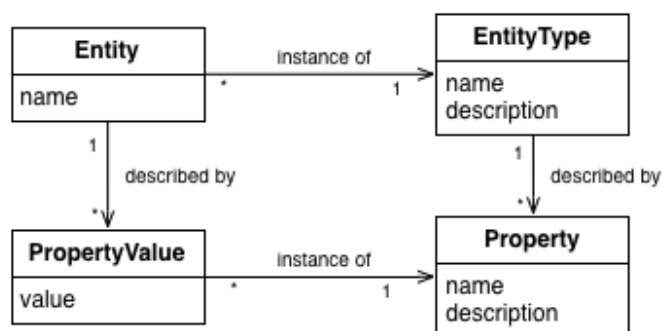


Figure 7 - Domain model for Knowledge Base

To be able to represent any part of the world, there is a need for a very generic and flexible data model that will allow any facet of the world to be represented in a Watch Knowledge Base. The most simple and powerful model found is by defining a generic representation called Entity, which is something that has a distinct and separate existence within the world, i.e. a representational unit (Figure 7). Every Entity belongs to an EntityType, i.e. a list of entities that can be grouped together because they have the same class or type and therefore share the same list of properties. A Property is a quality or attribute that an Entity of some EntityType can define and the definition of this Property to a certain Entity is a PropertyValue.

#### EntityType

It will serve the purpose of differentiating between the different facets of the world that one might want to watch and account for filtering known properties based on its type. It will carry out some metadata about the entities themselves. Examples of entity types might be domain objects such as TOOL, FORMAT, EXPERIMENT, POLICY, etc.

#### Entity

A domain object that represents an instance of entity that is of interest to the Preservation Watch component. Each Entity has one EntityType and thus a specific set of unique properties that describe it. Each Entity will have values for the properties that describe its EntityType. These values are measurements of a specific property in a point in time.

## Property

Each `Property` describes a certain “quality” of an entity. Each `Property` is specific for some entity type and only entities of that type can have values related to the specific property. This fact should make clear what types of questions can be proposed and will restrict the user of posing overcomplicated questions for that an automated answer cannot be provided. The `Property` object could also contain information about the units of the measured values and even some helper functions to convert values from one unit to another.

## PropertyValue

`PropertyValues` are atomic measurements of `Properties`. Each `PropertyValue` belongs to one specific property and one specific entity.

To better understand this model lets analyse an example, illustrated in Figure 8.

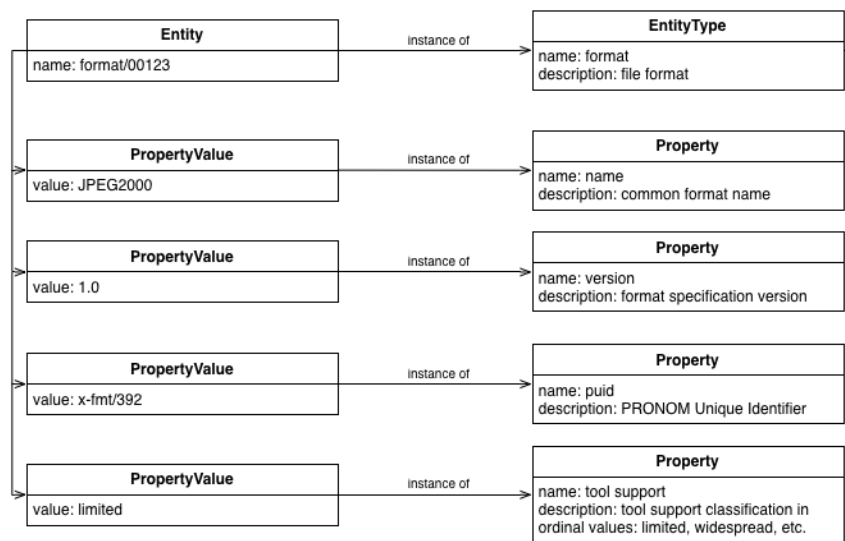


Figure 8 - Knowledge Base example

In Figure we have the `EntityType` “format”, which has a relationship with all `Entities` that represent a specific file format. We further define that every `Entity` format must (or should) define the properties: *name*, *version*, *puid* and *tool support*. The figure also defines a specific format, identified internally by the name *format/00123*, which has the common name JPEG 2000, the format specification version 1.0, the PRONOM Unique Identifier *x-fmt/392* and a *limited* tool support.

It should be noted that `EntityType` and `Property` have a different function than `Entity` and `PropertyValue` in the domain model. `EntityType` and `Property` allow defining what model of the world we accept, allowing for the normalization of information by the system administrators. Therefore, only system administrators will be allowed to change these two classes. The classes

Entity and PropertyValue have the actual information taken from the world, and are updated by the source adaptor via an interface defined in Section 3.2.5.

Having solely knowledge about the world is not enough to satisfy the needs of digital preservation and accountable decision-making that will use the knowledge gathered here for preservation planning. Information about how this knowledge was gathered is needed to ensure trustworthiness of the knowledge and, transitively, the decisions made. Therefore, there is a need to keep track of the provenance of the knowledge by defining what was its source and in what moment(s) the sources validate (by measurement) the property value.

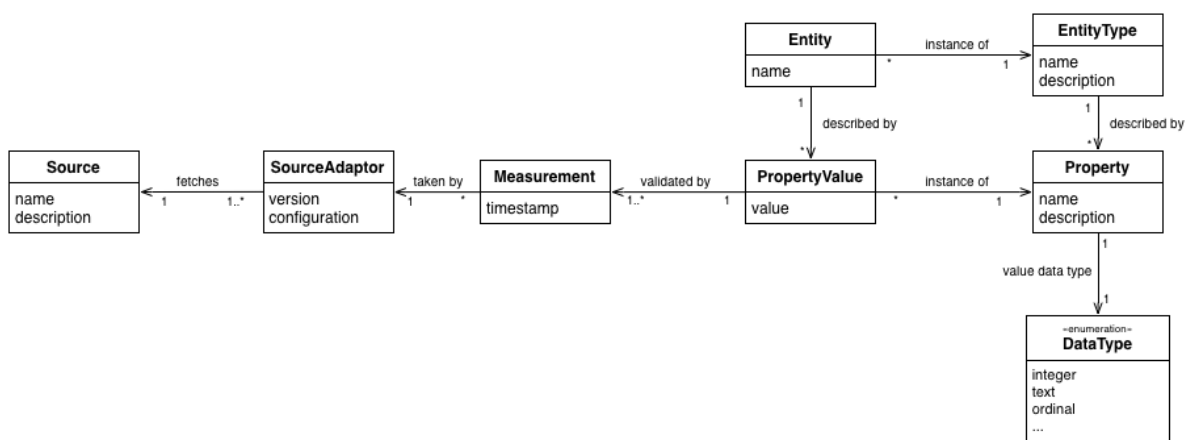


Figure 9 - KB with provenance information

Depicted in Figure 9 is the model for KB with provenance. Also, a Property defines some metadata about this PropertyValue such as, for example, the type of data of the PropertyValue (text, number, ordinal, etc.). This will allow defining restrictions on how the information is represented to make sure the values of the Property are normalized (all follow the same format), and can be understood by the question engine.

The data model must keep history of all the knowledge gathered, even if it was superseded by new knowledge. Being able to question the knowledge base as it was on a past date is an important feature to allow repeatability of the decision making process. Keeping record of how the world entities and properties evolve can be an important input to the decision process itself. The measurements have therefore two functions: 1) to keep the provenance of the information and also 2) to allow for keeping a history of the values taken, because when a source adaptor measures a new value, the old value and its measurements will be kept, simply by adding a new value related to the same entity and same property with the current measurement date. This means that an Entity will point to many PropertyValues that relate to the same Property, but will be able to distinguish what is the current one by getting the PropertyValue with relation with the most recent Measure timestamp. We note that the current Measure is not necessarily the most trustworthy one. The confidence and reliability of Measures is a concept of core importance. However, the information sources currently in scope do not explicitly provide confidence levels per Measure. We will thus consider introducing statically stored confidence as an inherent attribute of values after the first iteration of the Watch component. Collecting large numbers of measures from different sources



supports estimation of measurement confidence based on properties such as consensus, variance and convergence.

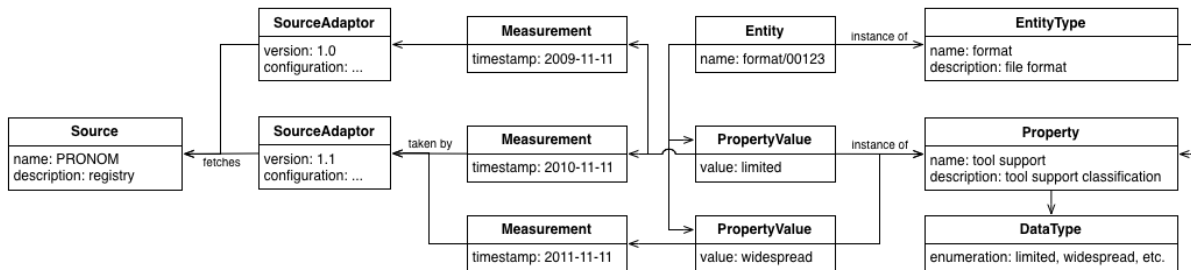


Figure 10 - Example of KB with provenance and history

In the example illustrated in Figure 10, the current value of the Property “tool support” of the Entity “format/00123” is “widespread” because that’s the PropertyValue related to the most recent Measurement of that Property, taken in 2011-11-11. In the same figure we can also see that the SourceAdaptor related to the same Source can be updated and new sources will be related to that SourceAdaptor.

With this domain model we are able to support answering the question “what is the current tool support of the format with the name JPEG 2000 and version 1.0”, with “widespread”, and also further justify the answer by saying “because that was the value measured by the PRONOM source adaptor version 1.1 at 2011-11-11”, which gives much more trustworthiness to our answer.

### 3.1.2 Watch Request

In this section we focus on the design of the Asynchronous Watch Request (Async Request) because the Synchronous Watch Request (Sync Request) is a sub-set problem of the Asynchronous one. In the final domain model the Sync Request will be identified.

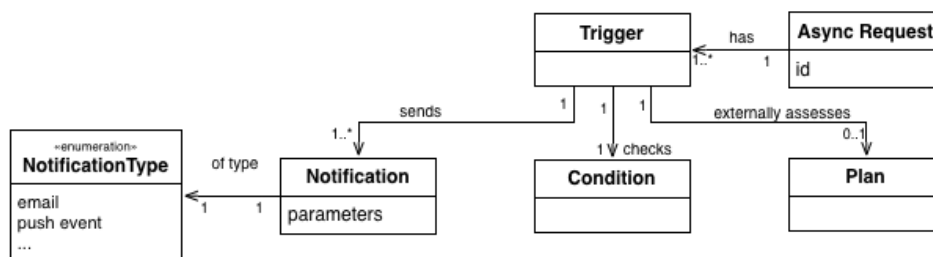


Figure 11 - Asynchronous Watch Request model (step 1)

Figure 11 depicts an AsyncRequest, which is a list of triggers. A Trigger constantly monitors a Condition and when the Condition check returns true, it sends a Notification. Optionally, and before sending the notification, the system can request an external component (e.g. Plato) to reassess a Plan, and only if the Plan result changes to send a Notification. The Notification defines the action that will be executed, which can be of different types (NotificationType),

e.g. send an email, or push the event to a certain API. The parameters of the action are defined on the Notification, and they can be, for example, the recipient of the email or the server that should receive the pushed event.

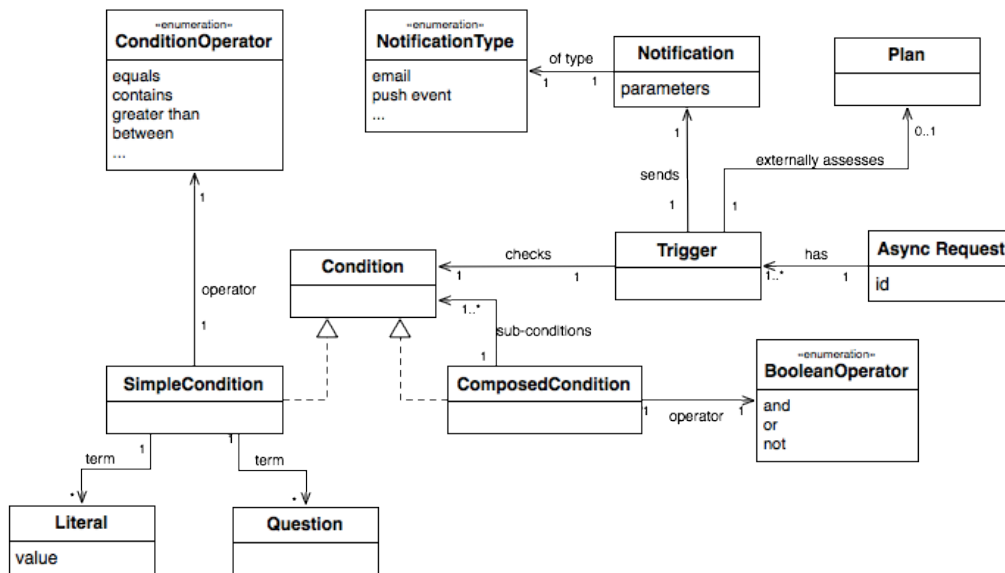


Figure 12 - Asynchronous Watch Request model (step 2)

Figure 12 further defines the model by introducing the Condition. A Condition can be of two types: a SimpleCondition that will be later explained, and a ComposedCondition that defines a Boolean expression with conditions. For example, suppose that C1, C2, C3 and C4 are simple conditions; a ComposedCondition can define “C1 and C2 or C3 but not C4”. The domain model represents this expression as a tree, allowing nesting of terms and complex Boolean expressions.

A SimpleCondition represents an expression that evaluates question results. The evaluation of this expression is a function that takes the result of a question and compares it with other question results or literals, returning true or false. The SimpleCondition uses condition operators and literals to define this expression:

- **ConditionOperator** is the comparison mathematical function used to evaluate the result of a question or composition of questions against a literal. Examples of operators: equals, contains, greater than, lesser than, etc.
- **Literal** is a constant and defined value against which the question result can be compared. It has a defined data type and must be compatible with the Condition Operator function. Examples of literals: 3, “Tiff”, True, etc.

Two examples, assuming that Q1 and Q2 are Questions and 1, 4 and ‘TIFF’ are literals, are “Q1 is between 1 and 4” and “Q2 contains ‘TIFF’”. Also, one can say “Q1 lesser than Q2”, comparing results of two questions in a single SimpleCondition.

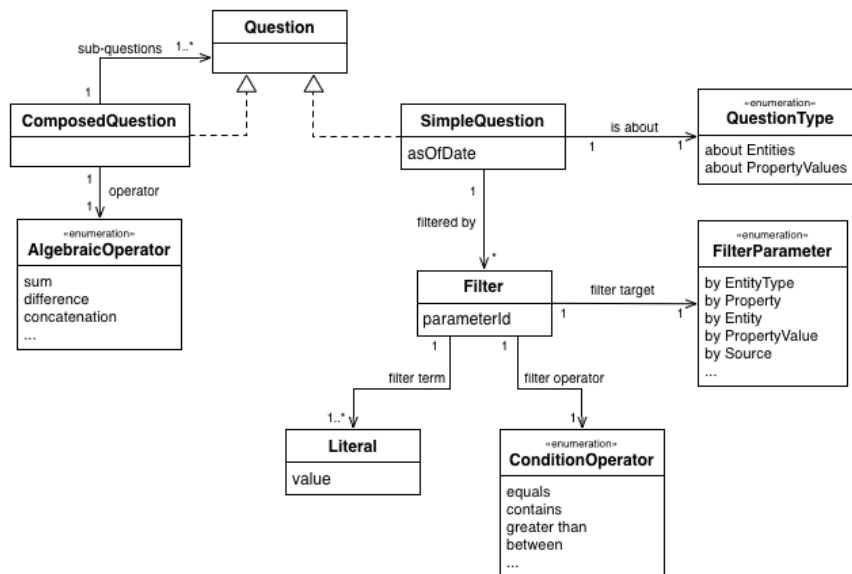


Figure 13 - Model of a Question

As illustrated in Figure 13, a Question can be simple or composed. A ComposedQuestion represents an expression that transforms the result on all sub-questions by using a function defined in AlgebraicOperator. For example, assume that Q1 and Q2 are SimpleQuestions that return a number, this model will allow us to represent “Q1 + Q2”, which can be valuable to represent Conditions like “Q1 + Q2 < 3”.

Based on the list of questions found in the previous sections we see that structure of the SimpleQuestions is very similar and can be divided into the following types:

“What are the values of properties X,Y,Z?”

“Are there any entities that have values for properties X,Y,Z?”

Based on these two types one can specify different conditions for the values of the properties of interest and thus tailor the questions to ones demands. The conditions can specify thresholds for the values or a specific entity for which the values should be provided. Thus more specific questions as in the examples above can be posed. Therefore, a SimpleQuestion is a query on the Knowledge Base information that can have the following return types:

- A Question about entities, returning a list of Entity names
- A Question about property values, returning a list of PropertyValue values

Whatever the type (or target) of the question, every SimpleQuestion can simply filter the response by its relationship with:

- EntityType
- Property
- Entity

- PropertyValue (specific property value, identifying the related Property in Filter's parameterId)
- SourceAdaptor
- Source

Therefore a Question has a type and a list of filters as defined above. Each Filter defines the FilterParameter, as listed above, the ConditionOperator and a Literal, already defined for simple conditions. Also, a Question must define the date that should be used to evaluate it. The default option will be the current date, but if we wish to compare the change of a result in time, we can compare the result of a Question in two different dates, the current and a past one.

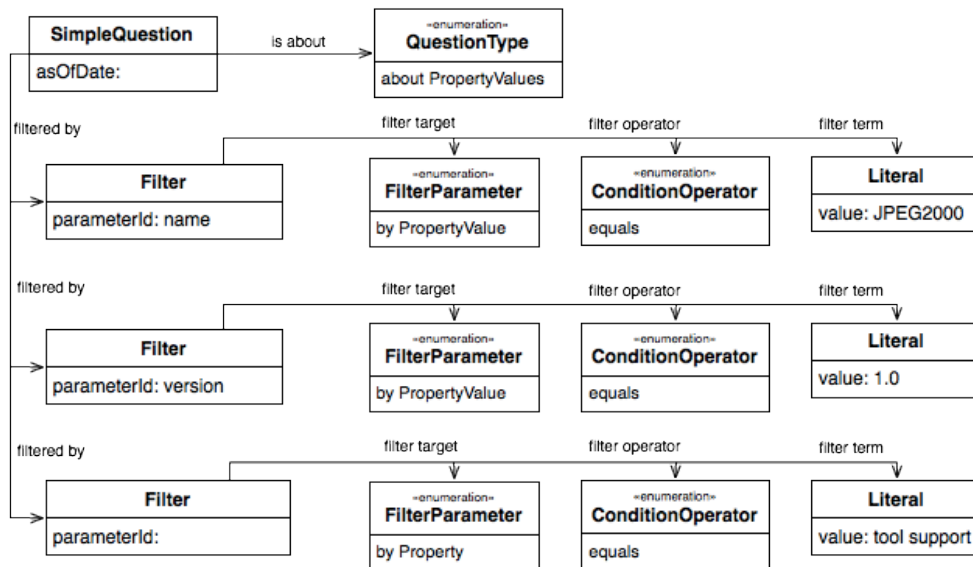


Figure 14 - Example of Question

Figure 14 depicts an example of Question that could be defined as “What is the tool support of the format with the name JPEG 2000 and version 1.0?” The Question is a SimpleQuestion of the type “about PropertyValue”. We are interested in the PropertyValue related to the Property “tool support” of entities which have a PropertyValue “name” equal to “JPEG2000” and the PropertyValue “version” equal to “1.0”. The result of this SimpleQuestion, assuming we have the Knowledge Base depicted in Figure 8, is a list with only the item “limited”. The result is a list and not a single value because some questions can give more than one value. For example, if we changed the question to “What is the tool support of the format with the name PDF?” and many versions of PDF existed in the Knowledge Base, then we would have a list of results.

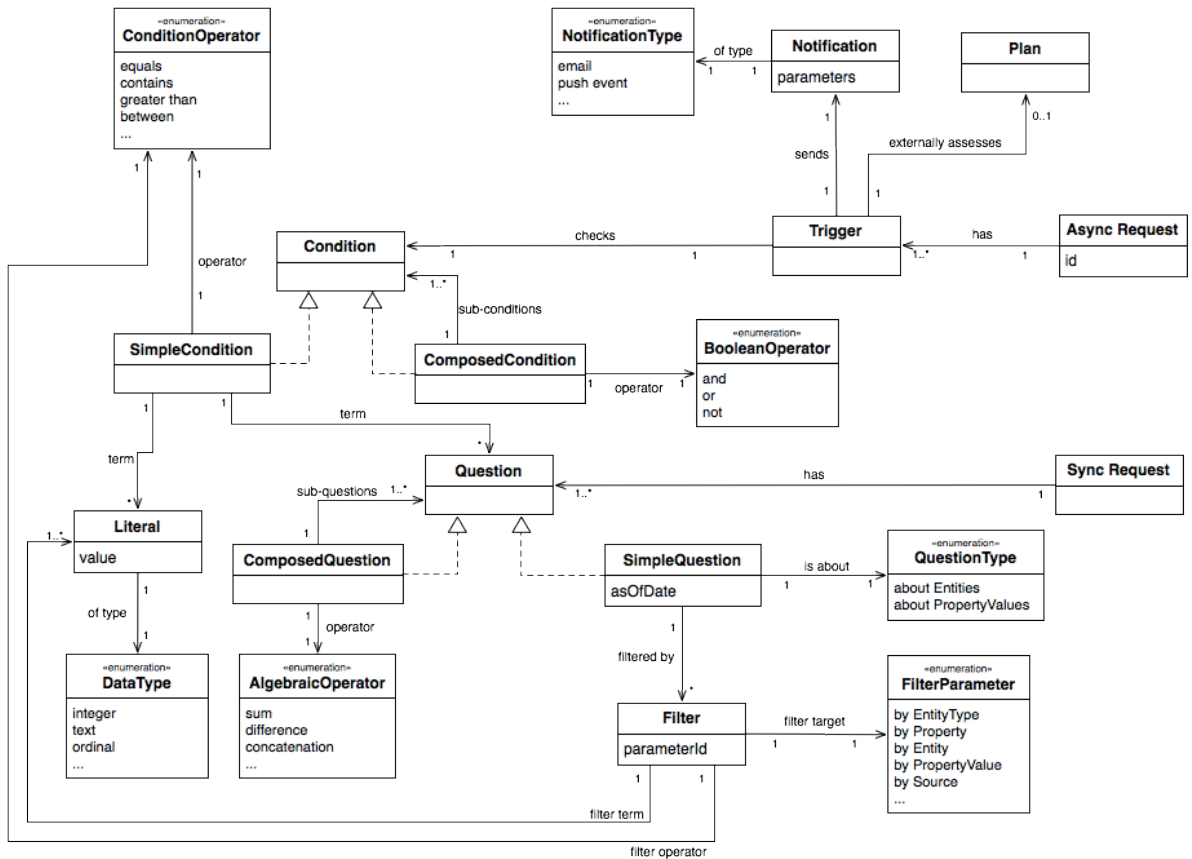


Figure 15 - Watch request model (final)

Figure 15 shows the complete Watch Request model, putting together conditions with questions, also adding the `DataType` information to the `Literal`, which can be useful for type checking operation arguments and compatibility of questions with the Knowledge Base.

### 3.1.3 Putting it all together

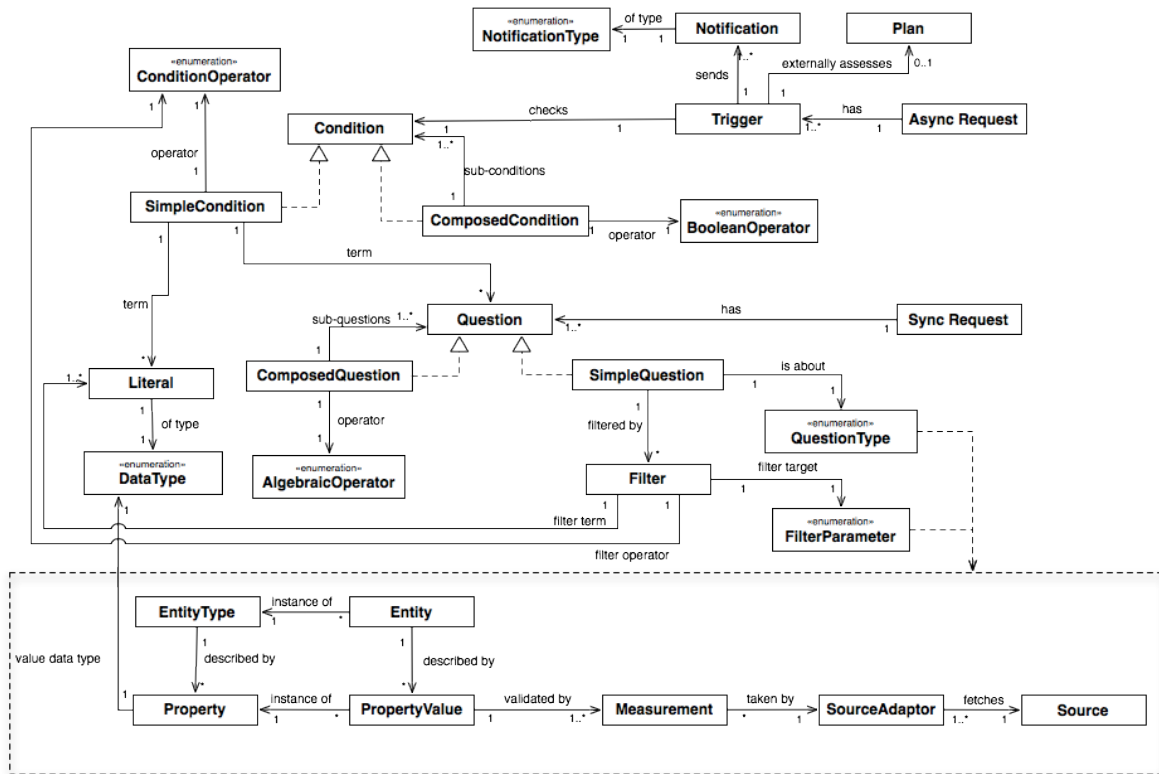


Figure 16 - Watch data model

When fitting together the model for the Knowledge Base and the Watch Request was found that the data type was present in both (relating to literals and properties), which allows the type checking of data and functions throughout the model (Figure 16). Another less binding point of contact is the `QuestionType` and `FilterParameter` that defines which class in the Knowledge Base part of the model some `Question` refers to or filters with.

## 3.2 Application Layer

This section describes the application layer and gives a more detailed overview of the sub-components. It specifies how all of them will fit together and communicate with each other, as well as the data flow between each of these.

### 3.2.1 Watch Central Service

The Watch Central Service acts as a central reference point to the Planner and therefore it incorporates some sub-components, such as the Assessment Service and the Notification Service, which are described in the following sections, but it also manages the Watch Requests and acts as a controller to the system, as it dispatches the data flow and the events throughout other components of the system.

The main task of the Watch Central Service is holding a list of Watch Requests and the responsibility of making them available as long as they are needed. It also dispatches important events, such as when a new value is stored to the Knowledge Base to its sub-components and passes the relevant

data for further processing. When the Assessment Service detects significant events, the central service uses the notification module to execute the triggers.

Figure 17 shows the high-level architecture of the Watch Central Service.

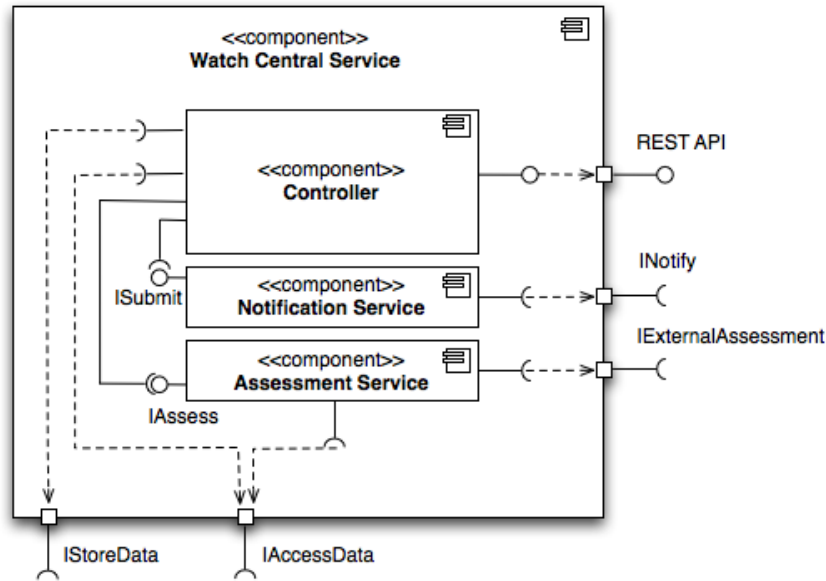


Figure 17 - The Watch Service subcomponents

The **Controller** is responsible for the Watch Request management and implements the REST API that is exposed to the clients. The API provides, at a minimum, methods for creating, reading, updating and deleting requests. Its responsibility is to make sure that each Watch Request is properly processed when it needs to be processed. This implies that it understands the difference between synchronous and asynchronous Watch Requests. It also has to make sure that Watch Requests are persisted as long as they are needed. Furthermore, the controller will hold a list of the Monitoring Services and allow their administration, e.g. enable, disable, pass different configuration to the corresponding source adaptors, etc. As the term administration implies, only specific users will be able to change the configuration of monitoring services.

The **Assessment Service** is responsible for evaluating Watch Requests with the current knowledge and provides the Controller with information, whether Triggers should be executed. More details about the Assessment Service are presented in section 3.2.2.

The **Notification Service** acts as a simple messenger and provides the needed facilities to call the clients back in their preferred way. As a beginning email and HTTP Requests (push notifications) will be implemented. More details are provided in section 3.2.3.

### 3.2.2 Assessment Service

The Assessment Service checks the new value(s) obtained by the source adaptors and decides based on the condition provided by the **Trigger**, whether such a significant event really occurred. Essentially, there are two types of conditions -- encoded simple conditions that can be evaluated locally, and complex conditions that need to be assessed by the decision-making capability.

Significance assessment first runs an **internal simple check** based on a local condition that can be evaluated without external knowledge (i.e. external to the Watch component). If this condition is met, an **external additional assessment** may be necessary to decide upon the significance of a change. If no external assessment was specified during the creation of the Watch Request, a **Significant Event** will be recognized. If an external assessment based on global knowledge is specified and available, it will be used. For example, the planning component can provide an assessment service that checks if an identified change leads to a potential change in preferences for existing plans. This can be done, simply by re-assessing the update objective evidence against the specified utility functions.

The assessment concept hence comprises two elements – an internal assessment based on local knowledge, where the assessment and decision making is done based on evaluation of the simple conditions of the trigger (as in the example), and an external assessment, one that is more sophisticated as it is based on global knowledge.

Note that the second type will not be part of the Watch component, as it will always require external knowledge. The Watch component itself cannot know the overlapping impact of other criteria, and the local conditions cannot be assumed to be independent across Watch Requests. The Watch component can use an assessment service in order to estimate the accumulated impact of more than one change events. The planning component will provide such a service for assessing the impact of monitoring conditions associated with plans, but manual assessment may also be possible and required in certain cases.

### 3.2.3 Notification Service

Generally, the Planner specifies the desired `NotificationType` when creating the Watch Request and its `Triggers`. At the beginning two types are going to be supported, email and Http Request (push notification). In the first case the user has to specify the recipient (optionally the sender) and in the latter case a web-address, an HTTP request type (GET, POST, etc.) and optionally parameters. In the latter case, for example, the Planner can setup a simple script that counts the GET requests on a URL and thus get information about each change of interest detected by Watch, or a more sophisticated call of a Web Service/REST Service can be setup as well. Note that these are just examples and no such services will be provided by the Watch component. Their implementation lies in the responsibility of the client.

As soon as the Watch Service decides that a `Trigger` has to be fired, the `Notification` of that `Trigger` is passed to the Notification Service via the `ISubmit` interface.

Since at this point it is unknown how many notifications the system will have to handle at a time, we assume that there might be a large number, so the suggested design tries to deal with this fact by using a queue that is responsible for the temporary persistence and the processing of the notifications. However, the first implementation might simplify these details and the queue might be just a dummy that sends the notifications sequentially without storing them. If the system requirements show that an optimization is needed, this change will affect only this sub-component implementation and the system will continue to work properly.

### 3.2.4 Monitoring Service

In general the Monitoring Services will be used as a layer of abstraction between the Source Adaptors and the Knowledge Base. This is necessary because in some cases information from more than one



source adaptor might be used in order to infer some new knowledge. In that sense, the monitoring services will serve the purpose of additional data pre-processing, if needed.

Otherwise their main responsibilities are to delegate the data for permanent storage, and more importantly to configure the source adaptors. The latter is done in order to fine tune the performance of the Source Adaptors.

### 3.2.5 Source Adaptors

A Source Adaptor is a component that gathers information from a Source and delivers it to the Watch component to be inserted in the Knowledge Base. This process can work in a *push* or *pull* model. The *push* model implies that a component running on the Source will send information to the Watch component. On the other hand the *pull* model implies that a component running on the Watch component will collect information from the Source using any protocol supported by the Source and pass that information to the Watch component in order to be inserted into the Knowledge Base.

As Figure 18 illustrates, the Watch sub-components relevant to the *Source Adaptors* are the Push API and IPullSourceAdaptor, IDelegateData and IConfigure interfaces. Any implementation of IPullSourceAdaptor collects information from the Source and delivers it to the IDelegateData. The Push API is a REST API that any Source can use to push information to the Watch component. The Push API also delivers the information to the IDelegateData interface that passes the information to the Monitoring Services that in turn inserts it into the Knowledge Base. The IConfigure interface is used by the Monitoring Services to configure the Source Adaptors.

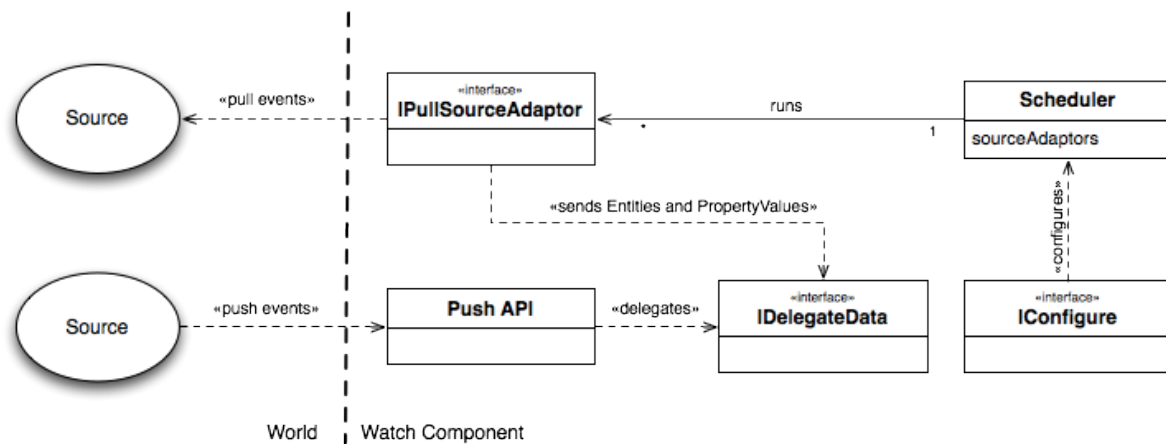


Figure 18 - SourceAdaptor model

The choice between the two models will depend on multiple factors, such as:

- Is the Source agnostic to the Watch component?
- Is it possible to create a software component that runs on the Source?
- Does the Source provides a network accessible API that can be used to retrieve data?
- Is the Source information public?

If the Source is agnostic to the Watch component, a *pull* Source Adaptor has to be used. The Source Adaptor would *pull* the information from the Source and insert it in the Watch component's

Knowledge Base. If the Source has a network accessible API, the Source Adaptor could use it to retrieve information; otherwise it would have to extract the information parsing the data in the format made available by the Source (e.g. HTML parsing). Furthermore, the information provided by the Source can be public or not. If it's public, the Source Adaptor can retrieve it without any type of constraints. On the other hand, if the information is restricted, a previous agreement should have been made with the Source owner that grants the Source Adaptor the right to extract the information.

On the other hand, if the Source is not agnostic to the Watch component and it's possible to create a software component that runs on the Source, a *push* Source Adaptor could be used. This Source Adaptor would send the desired information to the Watch component. The issues with the format of the information or its privacy would not be a problem with this type of adaptor because it's the adaptor that decides what information to send or not.

Independently of the adaptor's model (*pull* or *push*), the information exchanged between the Source and the Watch component is comprised of entities (`Entity`) and property values (`PropertyValue`). For any entity type containing a set of properties, an adaptor can create or modify instance entities or its property values. For each created or modified Entity or Property Value the responsible Source Adaptor and corresponding Source are also recorded, maintaining the required provenance information.

### **3.2.6 Web API**

The Web API is a simple consistent HTTP API that follows the REST architectural style as recommended by the SCAPE Technical Guidelines document. It enables client programs to communicate with and use the Watch component through the Watch Central Service and the `PushSourceAdaptor`.

In this document we do not provide the supported URLs, HTTP-requests and HTTP-parameters, as these will be designed during development. However, the APIs should provide clients with the possibility to submit synchronous and asynchronous Watch Requests, as well as edit (e.g. enable, disable) and delete these. Synchronous Watch Requests will be answered immediately via a HTTP response if possible. If this method turns out not to be feasible (due to HTTP timeout and longer processing time) the HTTP response might include a job id and a polling method will be provided for the client to fetch the results.

## **3.3 Front End Layer**

The front-end layer will be implemented as a web page user interface. Its sole purpose will be the administration and management of the data and watch requests. The Planners will be able to list the questions of the Watch component and create (change or delete) their own watch requests and thus specify their interest in some specific sets of Properties and Property Values.

Furthermore the Planners will be able to generate Triggers, which will be executed as soon as a significant event for that particular Planner occurs.

In order to make benefit of the internal knowledge base of the Watch component the user will be able to ask the current state of properties and their values via the web interface and thus retrieve important and relevant information in a synchronous fashion.

The following outlines a set of user stories that give more insight into the features, which the Watch component will provide.

Since the front-end layer is meant for human user agents and is represented by a traditional web application, in the following we refer to a Planner as a user.

### **User Login**

The user visits a web page and fills out a simple web form, providing his/hers username (email) and password. Afterwards the form is submitted via a click on a 'login'-button. If the provided credentials are correct, the user is redirected to a page that is referred as a dashboard-page in the following, where his/hers related information is visible. Otherwise an error is shown and the user stays on the same page. Furthermore, the login page has a link 'forgot your password', which provides a mechanism for password recovery.

### **List Properties**

The user lists the Properties of a specific Entity Type, in order to discover what information can be queried. To do this, the user selects the desired Entity Type from a predefined list (or combo box). For example, the user can filter by the Entity Type 'Format'. This will result in a small subset of Properties that each format might have (e.g. 'name', 'version', 'compression', etc.).

### **Create Watch Request**

From the dashboard the user can create a new Watch Request by clicking a link that redirects him to a new page with a standard form. The form might consist of a number of steps. The user provides the information, such as the Properties of interests by selecting them from a list. (This will be similar to the List Properties task). After selecting the desired Properties, additional filters and aggregate functions can be added. If the user wants to create an asynchronous Watch Request, then a Trigger must be added additionally. Otherwise the request will be evaluated immediately and a result will be shown (or an error if an erroneous request was submitted).

The Trigger consists of a Condition and a Notification. For these additional fields in the form have to be supplied such as the literal to compare the result with, the function to use while comparing (e.g. equals, bellow, above, etc.). Furthermore a notification type is to be chosen by selecting one from a predefined list (e.g. email, or push notification). Additional details such as recipient email have to be provided.

At the end the Watch Request is submitted by clicking a button. The user is redirected to the dashboard and gets a notification for the successful submission or stays on the same page and a list of errors is shown. An erroneous form submission does not delete the provided values of the form. The user might cancel the creation of the Watch Request at any time. This results in a redirect to the dashboard.

### **List users Watch Requests**

From the dashboard a use can see a summary of all Watch Request that he/she posed to the system. If one of these is selected, a detailed view is provided. Furthermore, for each Watch Request in the list, there is a button or a link to edit remove or disable/enable it. The edit button redirects to a form where the values of the Watch Request can be revised and stored. The delete button asks for confirmation and if agreed removes the Watch Request. The disable or enable Button changes the state of the Watch Request accordingly.

### **Modify Watch Request**

When a Watch Request is selected for modification, the user is redirected to a page with the same form as the create Watch Request form. Here the old values are already filled in the form and the user has the option to review and change them as needed. At the end the new values are stored by clicking on a 'Save' button and discarded by clicking on a 'Cancel' Button.

### **List old notifications**

From the dashboard the user can click on a link to list all notifications that were send out. The user gets redirected to a page where the notifications are displayed with some additional information such as what triggered the notification, when it was send out, etc.

### **Add/Edit/Remove known Property**

This feature will be usable only for a special kind of users, i.e. Admins. It will allow the extension of the knowledge base by adding a new Property for a specific Entity Type. In this case new Source Adaptors or new versions of Source Adaptors that can fetch this new information can provide it to the knowledge base.

In order to add a Property, the user clicks on a link and gets redirected to a page where a standard form is to be filled in. Edit and Remove will work similar as the Watch Requests features.

### **Add/Edit/Remove new known Entity Type**

This feature is also only for Admin users and allows the extension of the Entity Types. The user clicks on a button and fills out a standard form in order to create the new Entity Type. Edit and Remove are done in a similar way as described above.

### **Configure**

This feature would allow the Admin users to (re) configure the Source Adaptors used by the Monitoring Services. From the dashboard a list of all monitoring services can be obtained by selecting a link. The admin selects a monitoring service and receives a list with all related Source Adaptors. In order to configure one adaptor the user clicks on it and fills out a form that allows him to tweak the parameters of that specific Source Adaptor. At the end, the configuration is submitted or cancelled via a button click. If it is submitted the Source Adaptor has to reload its current configuration and start using the new one.

## **4 Technology Discussion**

This section clarifies the key technologies to be used in the development of the Watch component, discussing what are the most suitable technologies to fit the requirements of the data, application and front-end layer, and taking into consideration the system volume and throughput requirements and the SCAPE project global development guidelines.

### **4.1 Data Layer**

The data model as defined in section 3.1 is not very efficiently implemented in a relational data model. This happens because the model is very generic and open and must allow a flexible

representation of knowledge and at the same time a representation of questions about the knowledge. The relational model rigid specification is not adequate for this and makes the model quite complex and forcing the use of a very rigid query mechanism with reduced functionality to make knowledge inference.

A much more adequate model would be Linked Data, where many of the features would already be implemented and data representation could be simplified and more flexible because the ontological model if using a more generic and close to the required data model. These ontological databases already implement Boolean and Algebraic Logic and allow generic query systems with advanced features like transitivity, inversion and symmetry, which can be quite useful for the used questions.

Actually, using SPARQL as the representation format of Questions in the Watch API is a good option as it easier than more standardized than SQL (in all its flavours).

Using a RDF Triple-store may be more convenient for the representation of Questions, and maybe even more efficient than relational databases due to its structure (a fact to be further analysed), but some further analysis must be made in order to assess if a database of this type can scale to the volume of data that is expected for Watch to receive. Such analysis will be presented in section 4.2.

## 4.2 Volume estimation

In order to estimate storage and performance requirements, we estimate the expected volume of information to be collected. The main concern is to estimate the number of triples that would be stored in a linked data store. Linked data stores are still relatively new technology compared to relational databases, and scalability risks have to be detected early on (the datastore can present a bottleneck in the system if the read/write operation takes too long).

The estimation is based on a Monte Carlo simulation (Metropolis & Ulam, 1949) and estimates the number of triples after 36 months. This estimate thus projects the volume of knowledge in the Watch component one year after the SCAPE project finishes, with the component being active for about two years within the SCAPE project.

For each entity type three independent variables are defined: *entities*, *properties* and *property values*.

- *entities*: number of entities after 36 months
- *properties*: number of properties per entity after 36 months
- *property values*: how many time each property changed in 36 months (for example, the number 36 denotes that the property changed 36 times in 36 months, so on average one property value was collected every month.)

Variables are driven by a triangular distribution, so for each variable three numbers are estimated: *minimum*, *mode*, *maximum*. For example, for the number of entities in format we estimate the minimum number of entities expected after 36 months at 50, the most probable at 200 and the maximum at 5000. All these numbers can be estimated from the current technology status (for example number of entities in PRONOM registry). Table 11 provides the estimates for each variable and includes rationales for the estimates.

Table 11 - Variable estimations

Category	Estimates	min	mode	max
formats	entities	50,0	200,0	5000,0

	properties	15,0	25,0	50,0
	property values	1,0	6,0	36,0
	Minimum and maximum format estimates stem from the TB scenarios (i.e. content owners' collections) and PRONOM format. Minimum properties are derived from format properties as described in (Hamm & Becker, 2011), mode and maximum refer to PRONOM and P2 properties. We expect at least one value at the end of the timespan; a maximum frequency would be one value per month.			
action components	Entities	30,0	100,0	200,0
	Properties	20,0	40,0	100,0
	property values	1,0	12,0	36,0
	Entity estimates are derived from the expected content of the components catalogue, whereas properties estimates come from work package 10 and (Hamm & Becker, 2011).			
characterization components	entities	20,0	30,0	50,0
	properties	20,0	50,0	100,0
	property values	1,0	12,0	36,0
	Entity estimates are derived from the expected content of the components catalogue, whereas properties estimates are derived from (Hamm & Becker, 2011).			
QA components	entities	20,0	30,0	50,0
	properties	20,0	50,0	100,0
	property values	1,0	12,0	36,0
	Entity estimates are derived from the expected content of the components catalogue, whereas properties estimates are derived from (Hamm & Becker, 2011).			
experiments	entities	100,0	500,0	1000,0
	properties	50,0	100,0	200,0
	property values	1,0	1,0	1,0
	It is difficult to estimate the number of experiments of interest that will be conducted. In any case, each experiment constitutes its own set of values, hence the number of property values.			
collections	entities	20,0	100,0	200,0
	properties	10,0	50,0	100,0
	property values	1,0	6,0	36,0
	At a minimum, we would expect a collection profile for each set of data considered in SCAPE. Ideally, however, many more collections will be profiled.			
policy statements	entities	10,0	15,0	20,0
	properties	10,0	20,0	30,0
	property values	1,0	3,0	5,0
	The entities here refer to the expected number of categories of control policies in the policy model, and the properties refer to statements in these categories.			
simulation runs	entities	20,0	50,0	200,0
	properties	10,0	30,0	60,0
	property values	1,0	1,0	1,0
	We would expect a fair number of simulations to be relevant. The number of properties simulated will largely depend on the expressiveness of the simulation model. As with experiments, each simulation run produces distinct property values.			
repository operations	entities	5	10	20
	properties	20	30	50
	property values	1	6	36

We expect at least 5 repositories to be monitored at the end of the timespan.
---

From Table 11 is possible to calculate the approximate number of entries in a data store. For example, if the final result for the format would be *entities*=50, *properties*=15 and *property values*=1, that would result in total of  $(50*15*1)$  750 entries in a data store. Each of these entries describes a Measure taken by a Measurement and will be represented with more than one triple in a data store. Each Measure will likely be represented by the following triples:

1. Measure belongs to a Property
2. Measure describes an Entity
3. Measure has a Value
4. Measure is produced by a Measurement
5. Measurement has a Source
6. Measurement has a Timestamp

This estimation should not be taken as a final decision how each entry will be represented in a data store, but it is expected that the final number will be on the same order as this estimate. This means that for the example above (750 format entries) there would be  $(750*6) = 4500$  triples in a data store. It is important to note here that triples describing relations between entities and properties are also stored in a data store, but not included in the final estimation. These triples are neglected because they are constant (or change rarely) so they have no impact on the data store growth.

Estimated numbers (shown in Table 11) are fed into a Monte Carlo simulation, which is used to estimate the distribution of the final number of triples that are going to be stored in a data store. To ensure some level of statistical significance the simulation was done with 100 000 experiments (iteration runs). In each experiment the number of entities, properties and property values are randomly drawn from a triangular distribution where the minimum, mode and maximum are known (Table 11). After the numbers are generated final number of triples is calculated. This process is repeated 100 000 times. Thus the calculated results (number of triples after 36 months) are sampled<sup>13</sup> into 50 bins, and the distribution is calculated (Figure 19). In the Figure 19 the x-axis denotes bin sizes (in terms of number of triples) and the y-axis denotes number of experiments in each bin.

It is evident that the peak is between 3.000.000 and 5.000.000 triples. There is no value under 1.500.000, and the maximum value was under 45.000.000 of triples. The secondary axis shows the cumulative distribution. It shows the percentage of experiments that resulted with a value that is smaller than a certain value. It can be seen that around 90% of experiments ended with a result smaller than 13.000.000 of triples.

To have a better understanding of these results, we consult a well-known benchmark test, the Berlin SPARQL Benchmark (BSBM)<sup>14</sup>. Results are provided for the version 3 (February 2011)<sup>15</sup>. Testing was done on several link data stores (Virtuoso, BigOWLIM, 4store, BigData and Jena TDB) with 100 million and 200 million triples (which is an order of magnitude higher that we expect for the Watch data store). The hardware used was a single machine with processor Intel i7 950, 3.07 GHz and 24 GB

---

<sup>13</sup> for each value that denotes the final number of triples interval where it belongs to is found

<sup>14</sup> <http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/>

<sup>15</sup> <http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/results/V6/index.html>

RAM. The Focus was on three different values: loading, executing queries that only read from a data store, and executing a mix of queries that read and write to the data store. Results showed that even for data stores of 100M-200M triples, reasonable performance could be expected. For example, for fetching the data from 4store with 100M triples, a throughput of 117.6 queries per second is achieved. For the same datastore in case of inserting new triples the throughput of 20.3 queries per second is achieved, each adding 300 triples in a datastore. Even if the volume estimate proves to be false by a factor of 10, it still would make sense to use a triple store back-end, according to the Benchmark. In any case performance significantly depends on an implementation and query. Hence, further analysis will be needed before deciding for a concrete technical solution for deployment.

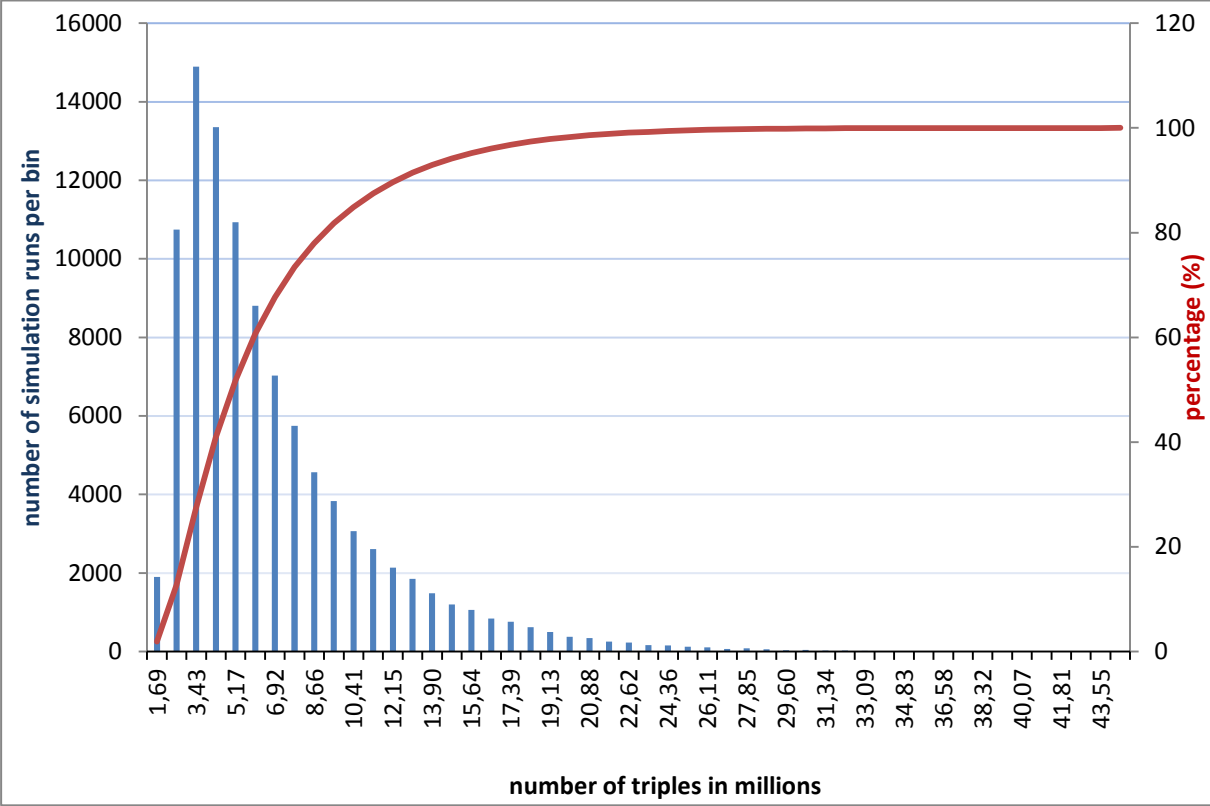


Figure 19 - Overall distribution of the simulation results

### 4.3 Application Layer

As the SCAPE Technical Guidelines recommend, we will use Java 1.6 as a programming language. Based on that requirement we have to choose frameworks that will support us and make the development easier. In any case we have to consider their long-term support and the level of experience the key developing partners have with them. Based on these facts we are left with the decision, whether the JEE Stack is to be used, or a different technology has to be chosen.

If the JEE Stack is chosen we have to consider an application server that supports everything that is needed, e.g. JPA, EJBs, JSF2, etc. Since we are going to develop a REST service and as the volume estimate shows a triple store will be a feasible solution it is implied that JPA and EJBs will probably not be needed. This allows picking up a server that is not JEE compliant. For now two choices are considered – Tomcat (v6) and JBoss (AS 7). The first one is more lightweight, easier to setup and probably sufficient if no specific features of the JEE Stack will be needed. Otherwise the second shall



be chosen, as it is fully JEE compliant. It is noteworthy to point out that the new version of the planning component will also use JBoss (AS 7), which would mean less performance issues, provided both services are deployed within the same instance/cluster.

Another possibility is a framework called The Play framework<sup>16</sup>, which offers a nice restful model and allows the rapid development of RESTful applications in Java with a lot of features that come out of the box such as Asynchronous Jobs, CRUD scaffolding and customization, etc. Since none of the developing partners has extensive experience with it, it might be pose a risk. However, it is not excluded that a simple prototype is developed in order to gain more understanding of the capabilities that the framework has. This is to be discussed.

#### 4.4 Front End Layer

The front-end layer can be decoupled from the application layer through the REST API. This might be problematic if huge amounts of content have to be delegated to the client, however, from the preliminary front-end feature scenarios this doesn't seem to be the case. Decoupling it will have the advantage that a different technology (not the same framework as the back-end) can be used. For instance Google Web Toolkit (GWT) might be used only for the front-end client side and in the back-end any other technology could be chosen. Other possible options are simple Java Server Pages, Java Server Faces or Java Rich Faces, or the aforementioned Play framework

## 5 Risks

This section shortly outlines the main risks foreseen with the development of the Watch component and associated services, and their associated mitigated strategies.

### **Sources may not provide enough information to draw useful conclusions and provide valuable information to the planners**

This would be a serious threat to the usefulness of the Watch component and has to be monitored within the project. By aligning the schedule of Watch services development and the development *and deployment* of key information sources such as the component catalogue, these risks can be kept under control. On the other hand, even pessimistic estimates point to a very useful level of information that will be collected; and by keeping the entire system open, the expectation for additional community involvement is not entirely unreasonable. The system also allows for the manual input of information. Using the knowledge of experts and the community and setting the right combination of incentives can always be a solution for relevant data acquisition. Finally, a key innovation of the Watch component is to *combine* information from different sources; this can certainly be achieved.

### **The division of work between the sub-components may lead to problems with interoperability and quality of the systems**

This risk will be mitigated by dependency decoupling, clear interface design, frequent meetings and the creation of mock up information sources that can be used to test other parts of the system.

---

<sup>16</sup> <http://playframework.org>

**The complexity and volume of the task may be too high to successfully deliver a useful prototype solution within the time span of the project**

As discussed before, the intention cannot be to develop a full-coverage automated monitoring and reasoning system for *any* aspect related to digital preservation. By focussing on feasible core aspects first, the complexity should be contained and the maximized benefit achieved. Furthermore, iterative development enables the team to react to additional insights about the complexity of certain aspects of information gathering.

**The amount of information may lead to performance degradation**

The amount of information that is expected to be inserted into the knowledge base and its rate of acquisition may lead to inefficiencies at the data level. However, triple stores have been tested extensively and have proven to be a solution that is able to cope with millions of triples. Additionally, certain triple stores have the capacity to scale horizontally resorting to clustering techniques, this being a way to mitigate potential risks of performance degradation.

**High complexity of the data model and API may hinder user adoption**

In order to use the Watch component, users will subscribe to notifications using a combination of questions about entities, properties and relationships between these. Those questions can be overwhelming even for very experienced users in a context where hundreds of entities and thousands of properties are available for consultation.

In order to mitigate this risk, the Planning component, whose development is closely connected to the development of the Watch component, will be able to suggest or even directly subscribe notifications on part of the end-user.

Additionally, the system may be enhanced with the functionality to present pre-defined notifications that have been uploaded by other users or by planning experts.

## **6 Summary and Outlook**

### **6.1 Summary**

Monitoring the preservation environment is a crucial part of the long-term viability of a system and the data it preserves. Although the monitoring function is a separate capability of a system, it is inseparable from the decision-making function, supporting the planning process with continuous tracking of the suitability of the decisions and delegating the risk assessment of the perceived significant changes back to planning.

Such a system is essential for digital preservation, but using manual processes to track all the environment variables that might affect the multitude of object file formats within a repository, with all their different characteristics and contexts, is not feasible. Currently, no tool or service exists that could properly provide this function.

This document delineates the design and development of such a system, named the Watch component, based on the knowledge of past research and experience and going forward by defining new concepts and strategies. The Watch component is aligned with the Planets Functional Model, but delegates some of the functions defined there as part of the "Watch Function" to the decision-making component of Planning. This new interdependency between watch and planning becomes a

new approach that tries to alleviate the burden of performing risk assessment within the Watch component and also avoiding depending on external sources for this task.

Based on refined requirements, real-world scenarios and the possible sources of information, an architecture is presented defining a data model to represent the world's interesting properties in an internal knowledge base and also to represent questions about these properties. The architecture also defines a set of services that allow, from one side, the inspection and retrieval of information from the outside world that can be inserted in the internal knowledge base, and from the other side, the query of information, definition of event significance and subscription of notifications (e.g. email).

To enable the structuring of information, the Knowledge Base data model must define two sets of elements. One, managed administratively, describes what model of the world is accepted by defining what type of entities can exist and what properties may be defined for them, and another, managed by the sources of information, describes instances of these entities and the values of their properties. The data model also keeps a register of the information provenance and history of changes to allow the traceability of information, which will transitively improve the traceability of the decision making process.

For the sources part, a set of interfaces is defined to allow pulling information from the outside world, specifically used when the relevant sources remain agnostic to the watch service. These also serve to push information into the Watch component, used for example when sources of information need more control over what information is sent and how. Also, a manual web user interface empowers users to send information when no automatic source is available.

The technologies needed to develop this system can be analysed in accordance with the three-layer model view: data layer, application layer and front-end layer. The data layer must allow the flexible model described for the knowledge base and provide the query features needed to answer questions about the world. The most appropriate technology for the representation of a domain model that is so close to an ontology is clearly Linked Data implemented as a Triplestore. Scalability issues of Linked Data were analysed, and it was concluded that the current state of the art on Triplestores performance is more than adequate for the estimated volume and throughput needed by the Watch component data layer. The technologies to be used in the application and front-end layer are less restricted from the problem side, but should follow the guidelines defined in the SCAPE project with preference for the ones with which the development team has experience with.

## **6.2 Next Steps**

The next phase is planning of the development tasks, which will entail defining at a finer grain which technologies to use based on further analysis and the experience of the development team. The development will strive for an early prototype that would allow the evaluation of the suitability of the defined interfaces and the interoperability with other components, specifically with the planning component. The prototype will also serve as a proof of concept of the suitability of the underlying technologies and the ability to represent the outside world information and allow the posing of questions and creation of triggers and notifications.

Beyond the prototype, the Watch component will be further developed to integrate with all sources of information needed by all identified questions. This includes the development of a repository watch adaptor that allows monitoring of repository actions, content and usage, and the development of a web content adaptor that will integrate with web archives and allow the monitoring of the

content of the web, identifying trends and analysing the renderability of pages on newer browser versions.

The information gathered by the Watch component will also enable inferring future behaviour of repositories and global trends and simulating the effects of planning and watch. This simulation tool will allow, for example, computing the expected state of a repository at a certain time in the future, enabling a forward-looking watch perspective that proactively monitors expected developments.

## 7 References

Antunes, G., Becker, C., Borbinha, J., Proença, D., & Vieira, R. (2011). *Shaman reference architecture (version 3.0)*. SHAMAN project report. Retrieved from <http://shaman-ip.eu>

Becker, C., & Rauber, A. (2011). *Four cases , three solutions : Preservation plans for images*. Vienna University of Technology, Austria. Retrieved from <http://www.ifs.tuwien.ac.at/~becker/pubs/becker-four2011.pdf>

Becker, C., Antunes, G., Barateiro, J., Vieira, R., & Borbinha, J. (2011). Control Objectives for DP: Digital Preservation as an Integrated Part of IT Governance. *ASIST 2011*. New Orleans, USA. Retrieved from [http://www.ifs.tuwien.ac.at/~becker/pubs/becker\\_asist2011.pdf](http://www.ifs.tuwien.ac.at/~becker/pubs/becker_asist2011.pdf)

Becker, C., Kulovits, H., Guttentbrunner, M., Strodl, S., Rauber, A., & Hofman, H. (2009). Systematic planning for digital preservation: evaluating potential strategies and building preservation plans. *IJDL*, 10, 133-157. doi:10.1007/s00799-009-0057-1

Beer, S. (1981). *Brain of the Firm* (2nd ed., Vol. 1, pp. 181-182). John Wiley & Sons.

CCSDS. (2002). *Reference Model for an Open Archival Information System (OAIS)*. Retrieved from <http://public.ccsds.org/publications/archive/650x0b1.pdf>

Ferreira, M. (2009). *Preservação de longa duração de Informação Digital no contexto de um Arquivo Histórico*. Universidade do Minho. Retrieved from <http://hdl.handle.net/1822/9563>

Hamm, M., & Becker, C. (2011). *Report on decision factors and their influence on planning*. SCAPE Project Deliverable (D14.1). Retrieved from <http://www.scape-project.eu>

International Council on Archives. (2000). *ISAD(G): General International Standard Archival Description*. Ottawa. Retrieved from [http://www.icacds.org.uk/eng/ISAD\(G\).pdf](http://www.icacds.org.uk/eng/ISAD(G).pdf)

Lawrence, G. W., Kehoe, W. R., Rieger, O. Y., Walters, W. H., & Kenney, A. R. (2000). *Risk Management of Digital Information: A File Format Investigation*. *The Journal of Academic Librarianship* (Vol. 27, p. 75). Washington D.C.: Council on Library and Information Resources. doi:10.1016/S0099-1333(01)00249-X

Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, 44(247), 335-341. Retrieved from <http://www.amstat.org/misc/TheMonteCarloMethod.pdf>

- OECD. (1996). *The OECD Jobs Strategy: Vol.1: Technology, Productivity and Job Creation* (p. 282). Organization for Economic Co-operation and Development. Retrieved from <http://www.amazon.com/OECD-Jobs-Strategy-Technology-Productivity/dp/9264148817>
- Pearson, D., & Walker, M. (2007). *Report of the Format Notification and Obsolescence Service (AONS II)*. Retrieved from <http://www.apsr.edu.au/aons2/report.pdf>
- Portuguese Institute for Quality. (2007). *NP 4457:2007 - Portuguese standard for Research, Development and Innovation Management - Management System Requisites*.
- Roberts, B., & Swirrl IT Ltd. (2011). *A New Registry for Digital Preservation - Conceptual Overview*. Open Planets Foundation. Retrieved from <http://www.openplanetsfoundation.org/new-registry-digital-preservation-conceptual-overview>
- Sierman, B., & Wheatley, P. (2009). *Report on the Planets Functional Model*. Planets project deliverable (PP7-D3-4). Retrieved from [http://www.planets-project.eu/docs/reports/Planets\\_PP7-D3-4\\_ReportOnThePlanetsFunctionalModel.pdf](http://www.planets-project.eu/docs/reports/Planets_PP7-D3-4_ReportOnThePlanetsFunctionalModel.pdf)
- Sierman, B., & Wheatley, P. (2010). *Evaluation of Preservation Planning within OAIS, based on the Planets Functional Model*. Planets project deliverable (PP7-D6.1). Retrieved from [http://www.planets-project.eu/docs/reports/Planets\\_PP7-D6\\_EvaluationOfPPWithinOAIS.pdf](http://www.planets-project.eu/docs/reports/Planets_PP7-D6_EvaluationOfPPWithinOAIS.pdf)
- Stanescu, A. (2005). Assessing the durability of formats in a digital preservation environment: The INFORM methodology. *OCLC Systems & Services*, 21(1), 61-81. Retrieved from <http://webdoc.sub.gwdg.de/edoc/aw/d-lib/dlib/november04/stanescu/11stanescu.html>
- The Open Group. (2009). *TOGAF version 9: The Open Group Architecture Framework*. The Open Group.