Demonstration Epimenides

www.ics.forth.gr/isl/epimenides/

Users' Guide

Last Update: 18/2/2014
Author: Kargakis Yannis (FORTH)

1. About

This demo application aims at demonstrating the approach proposed in the paper titled: "Conversion and Emulation-aware Dependency Reasoning for Curation Services", in the context of APARSEN WP25, Task : Interoperability Strategies.

It is a web application that checks the performability of a task over an uploaded Digital Object. The system identifies the dependencies that are missing for performing a task on the uploaded Digital Objects.[1]

2. Menu Options

The menu of the application, as you can see in the next Figure, is separated in 3 sections. The first contains the main option of the application: "Upload Digital Object".

[1] The interested user could read the following articles for a better understanding of the objectives, the implementation and the architecture of this application :

- **Conversion and Emulation-aware Dependency Reasoning for Curation Services**
  Proceedings of the 9th Annual International Conference on Digital Preservation (iPres2012), Oct. 2012, Toronto.

- **APARSEN D25.2 Interoperability Strategies**

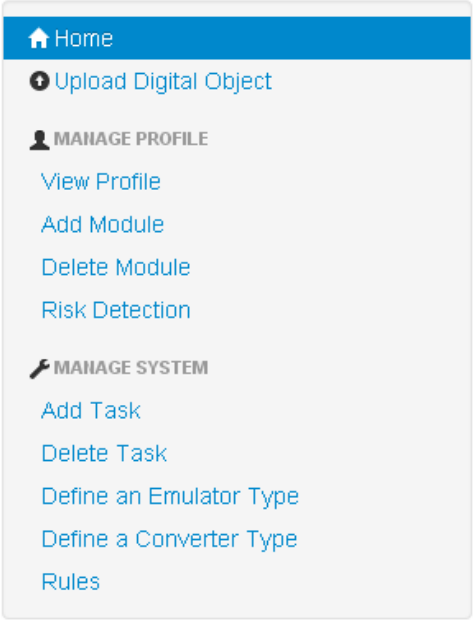

Figure 1 Menu Options

The “MANAGE PROFILE” section contains options available to any user (simple users). The user can add/delete modules to his profile, where a profile is a list of modules.

The “MANAGE SYSTEM” section contains options for a curator user. Such a user has also the ability to define Tasks, Emulators and Converters. To properly add a Tasks/Emulator/Converter one has to provide extra information from which the application will produce the required rules. A simple user can add to his profile an emulator X, only if it has been properly defined from a curator user (and consequently the application has produced the required rules).

3. Usage Scenario

Here we describe a simple usage scenario of this application. The user uploads a digital object in order to check if he/she can perform a certain task assuming his profile (e.g. in his computer) on this object. A profile is the set of modules that are assumed to be known (available or intelligible) by the user. A module can be a software/hardware component or even a knowledge base expressed either formally or informally, explicitly or tacitly, that we want to preserve.

The user explores the dependencies involved in the task and finds those that are missing. The user can add the modules that (s)he already has in order to fill the gaps, and this is actually the method for defining his profile gradually.

The system currently supports a few common tasks, like rendering for documents/images and runnability for software. The user can create and check the performability on a digital object, of his own tasks.

3.1 Check Performability

Follow the next steps:

a) In the login screen select “Just Run a Demo” (Demo User) to load a demo profile (super user). This profile contains the following modules (as also you can see in “View Profile” option) :

Module Name	Module Type	Added By
notepad	TextEditor	Yannis
windows7	WinOS	auser

b) Upload some digital objects:
Select to load the “demo zip 1” (Figure 2) file that contains 5 digital objects

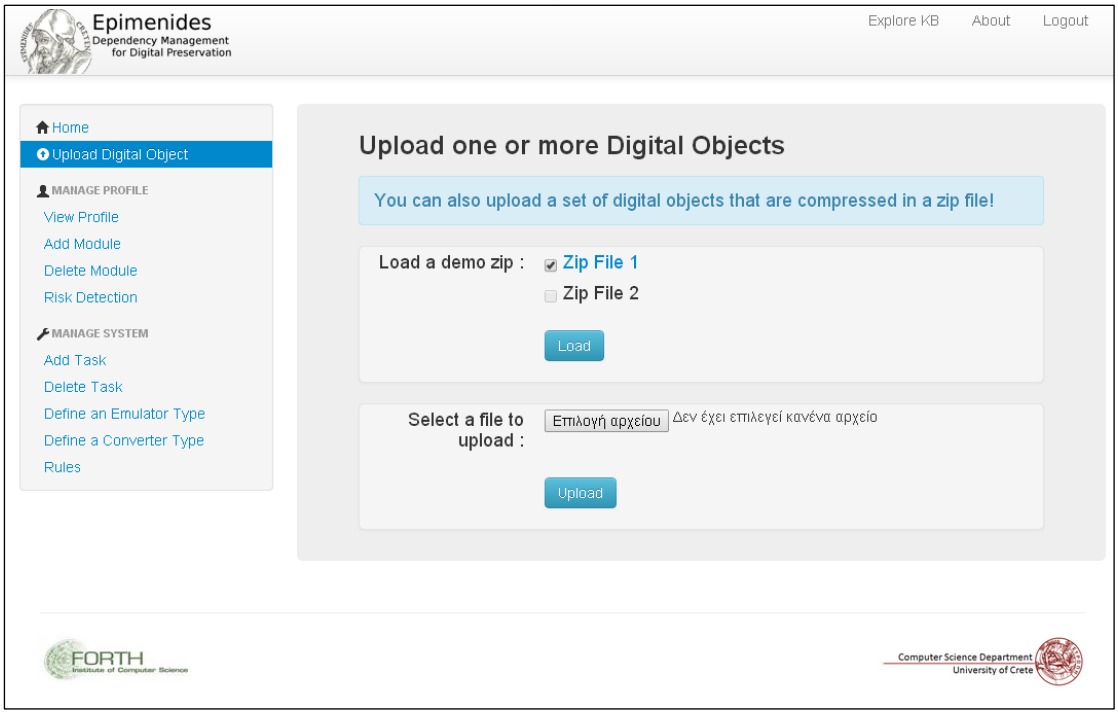

Figure 2 Upload Digital Objects

c) For each one of the digital objects the system has recognized the object type (Figure 3) and the applicable tasks (Figure 4). You can use the “Select Files and Task to Analyze” button to select the 3 digital objects that can be analyzed for a task.

Currently the system contains the following tasks :

- “Readability”
- “Runnability”
- “Rendering”

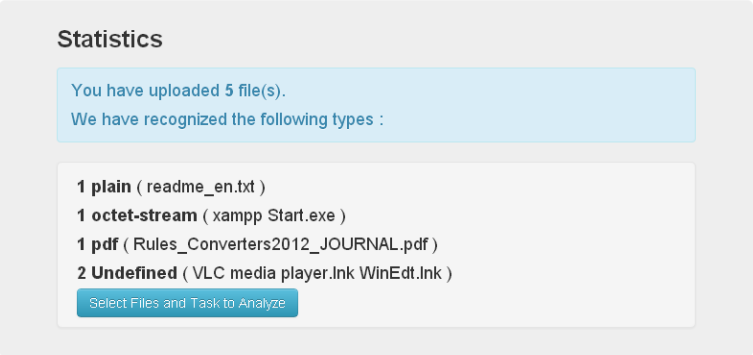

Figure 3 Epimenides has recognized the Mime Types

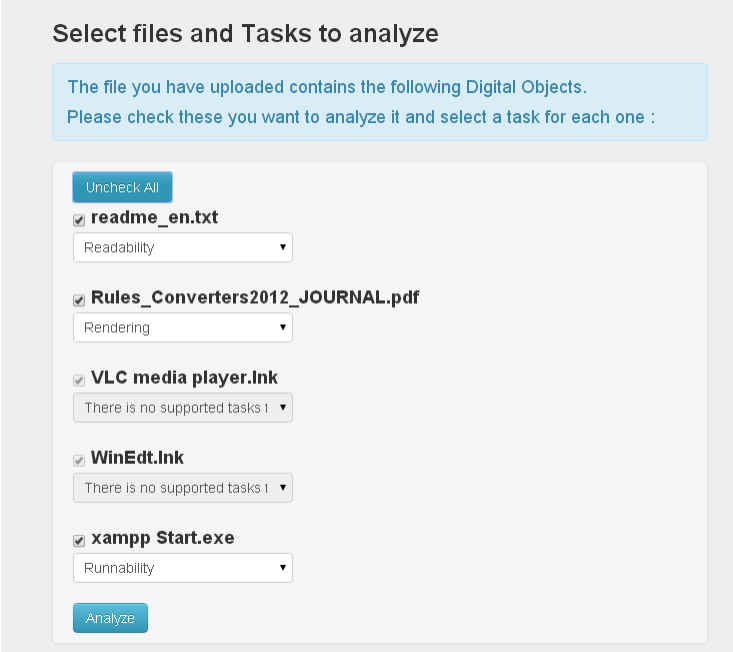

Figure 4 System finds the tasks that usually make sense to apply to the uploaded digital objects

d) In the next screen (Figure 5) you can see the tasks that can be performed. You can explore the dependencies for each one of the digital objects, for

example select to explore the ".pdf" file whose rendering cannot be performed (this is why is colored red), in order to see the dependencies that are missing and are required for performing the selected task.

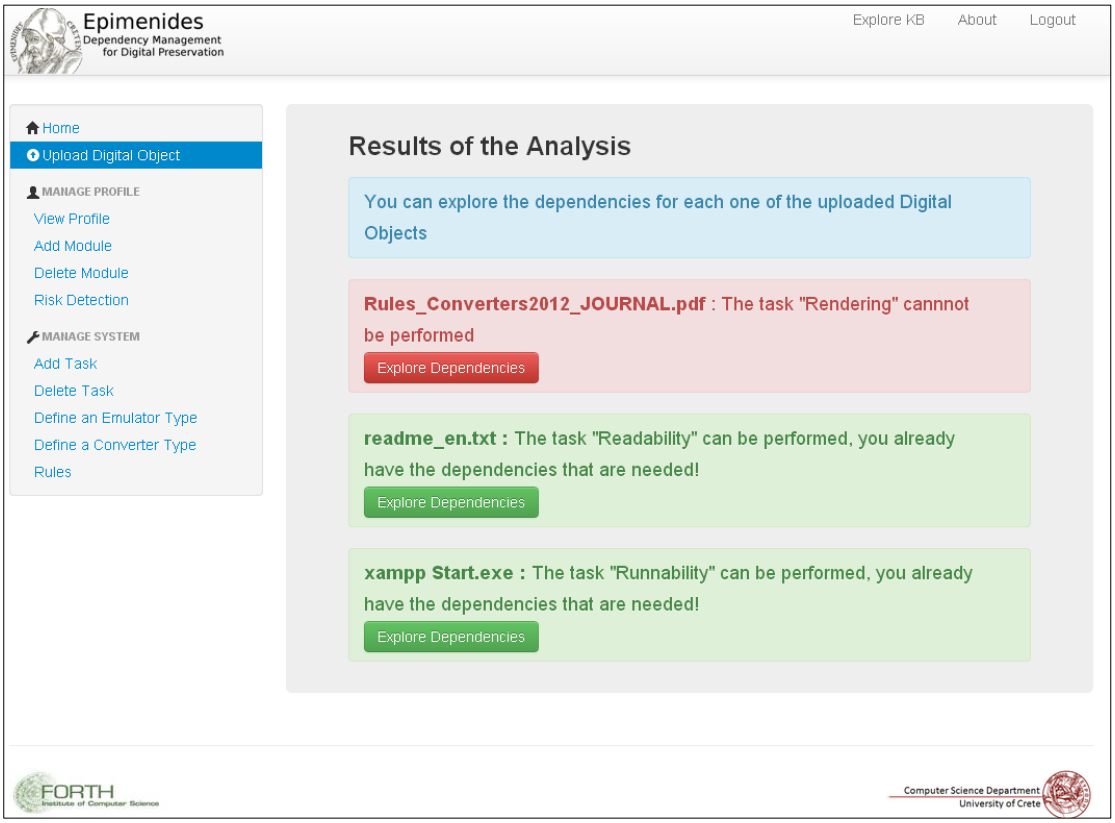

Figure 5 Results of the analysis

e) As you can see (Figure 6) a dependency for the task "Rendering" is : Rendable(x,y) : -pdf(x), PDFViewer(y)

Figure 6 Explore Dependencies of a Task

f) Select “Add Module” and add a module with type : “PDFViewer" (Figure 7). Upload again the digital objects (steps b - d). As you can see now all the selected tasks on the digital objects can be performed!

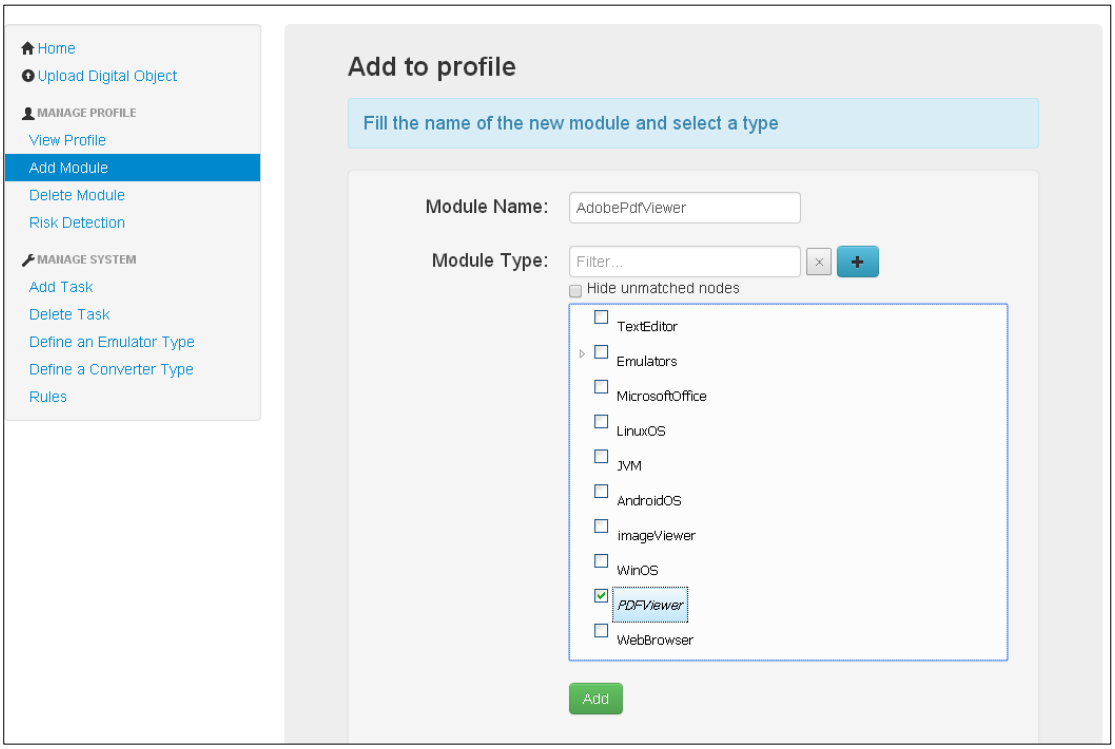

Figure 7 Register a new Module

g) Select “Delete Module”, delete the module “windows7”. Upload again the digital objects (steps b - d). Now the task “Runnability” cannot be performed as the windows operation system is missing.

h) Select “Add Module”, add an android OS:

Module Name : *“android4.0.2”,* Module Type*: “AndroidOS"*

and an Android to Windows Emulator (this emulator has already been defined):

Module Name : *“AtoW”,* Module Type : *Emulators/EmulatorAndToWin”*

Upload again the digital objects (steps b - d). Now the task “Runnability” can be performed due to the emulator!

3.2 Adding Tasks

Finally a curator user can add his own tasks. He should provide some input and the application produces the required rules and this actually will clarify the dependencies of the new task.

For example, select the option “Add Task” and add a task with :

i. performability name : *“Edit”*

ii. Applied In : “*text/plain”*

iii. Module Type : *“Text Editor”*

Upload again the digital objects, in step c now we can select the task “Edidability” that we have just created for the .txt file.

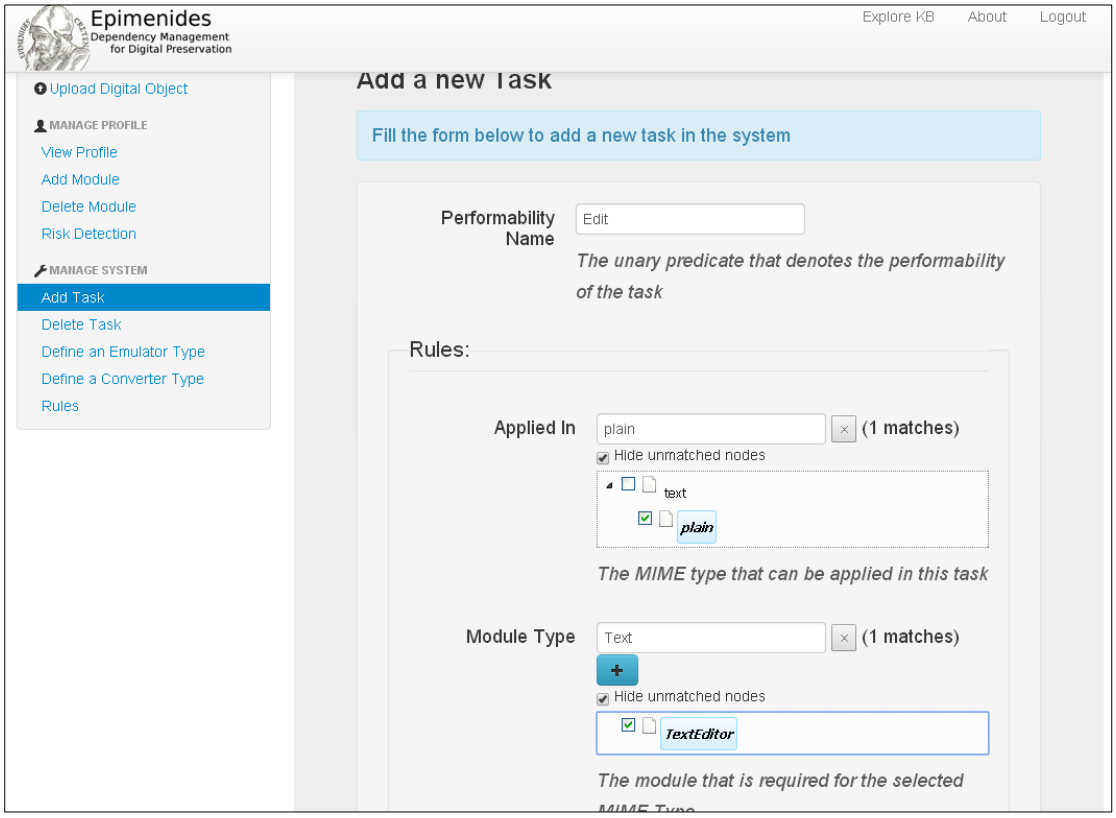

Figure 8 Define a new Task