Datapkg-gui

Daniel Graziotin <d at danielgraziotin dot it> - http://task3.cc
SoNet research unit at FBK

Abstract

The Open Knowledge Foundation [1] welcomes many projects regarding the possibility to open data and enable free knowledge sharing around the world. One of these projects is Ckan / The Data Hub[2], that is a registry of open knowledge datasets. See the related websites for more information. OKFN and open-source developers created a Python tool built around The Data Hub APIs, called datapkg[3]. Datapkg is able to query The Data Hub in order to search, obtain, create and redistribute packages of knowledge. What is currently missing is a graphical user interface for datapkg, that will help non-proficient people to use the tool and enlarge the user base of Ckan services. This project will create such GUI. Additionally, the program will be coded to either be run stand-alone or as a component of the SOFA Statistics[4] tool, therefore allowing SOFA users to directly perform statistical calculations on The Data Hub data packages.
The whole experience is a 10 weeks solo project performed at the Fondazione Bruno Kessler[5], as part of a University stage.

About this document

This paper acts both as a journal of thoughts/activities and as a report of the project. It will be incrementally updated. The first version of this document has been released on 2011-10-06.

Glossary Terms

Name	Aliases	Documentation
Ckan		CKAN is the Comprehensive Knowledge Archive Network, a registry of open knowledge datasets and projects (and a few closed ones). CKAN makes it easy to find, share and reuse open content and data, especially in ways that are machine automatable. CKAN combines the features of a listing/registry, a dataset index and a wiki. As a registry it acts like Freshmeat but for open data And content resources. However it adds to a simple registry in key ways.
Open Knowledge	OK	Content/Data/Information (Genes to Geodata, Stats to Sonnets) Open = Freedom to Access / Use / Re-use / Redistribute
Dataset	Package Data Package	the object representing datasets in CKAN and, as such, Is the central domain object.
Resource		A Resource corresponds to a file, API or other online data resource. A Resource is associated to a Data Domain Model/Package (which may have several Resources).
Group		provide a way to create a curated collection of datasets.
Relationship		CKAN allows you to define relationships between data packages.
Spec	Specification	A String identifying a package: ckan://{package-name} file://{index-path}/{package-name} db://{packag-name}
Index		A wrapper for interacting with Repositories of packets.
Repository		a storage location from which a package may be retrieved.
Metadata		Information about a Dataset. There exist a core metadata (that is, mandatory metadata) and extra metadata (arbitrarily defined)
Manifest		A list of resource(s) the package provides.

1 http://www.okfn.org
2 http://thedatahub.org
3 http://okfn.org/projects/datapkg/
4 http://sofastatistics.com/
5 http://fbk.eu

Datapkg

Datapkg Complexity

As pointed out in this blog entry[6], the first problem encountered has been in the internal structure of datapkg. The code is well-written and well organized, but the design decisions are poorly documented. It is difficult to build something around it. Therefore, we took the decision of perform a design study and documentation of datapkg organization. The results are published in this section. Please note that the following UML diagrams do not cover the whole datapkg project. They have been used in order to better understand datapkg internal structures and behaviors.

Datapkg Use Cases

The first useful UML diagram is a use case diagram that shows the main functionalities of the tool:

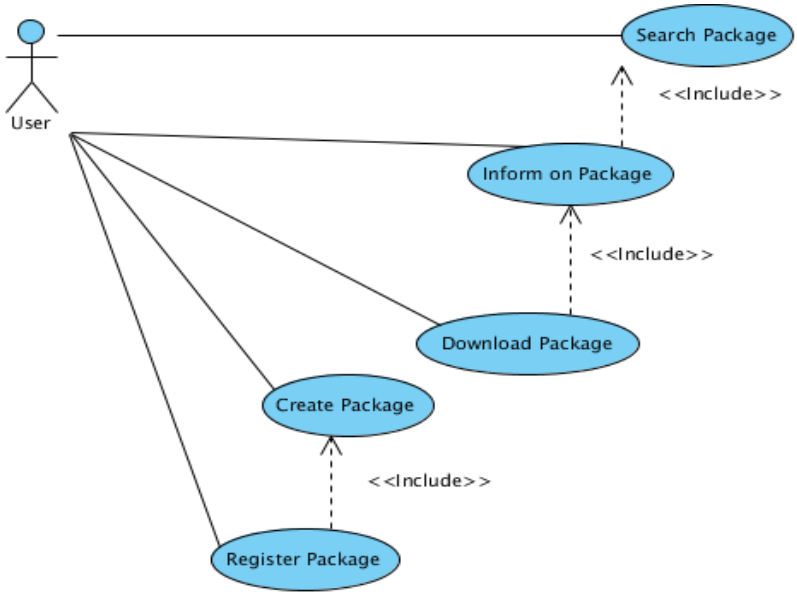

Illustration 1: datapkg use cases

These five basic actions are the most common and are useful both for learning to use the command-line utility and the first actions that the GUI will perform, too. They are better investigated in the study of the components of the system.

Datapkg Components

Datapkg has the code divided in modules and packages. Here we present a simplified hybrid model that combines UML class diagrams and Python modules plus packages. In Illustration 2 we represent the content of the main package of datapkg. The classes have some useful methods reported and the basic relationships among them are represented.
In Illustration 3 we have a focus on the Index package, that contains the various indexes used for each repository. It is important to understand indexes because they are responsible for performing operations on packages.
Illustration 4 represents the contents of the util package. It is moslty used for effectively downloading resources.
Illustration 5 is a representation of the command line classes. It reflects the use case diagram, as expected.

6 http://task3.cc/343/first-impressions-of-datapkg-how-to-proceed/

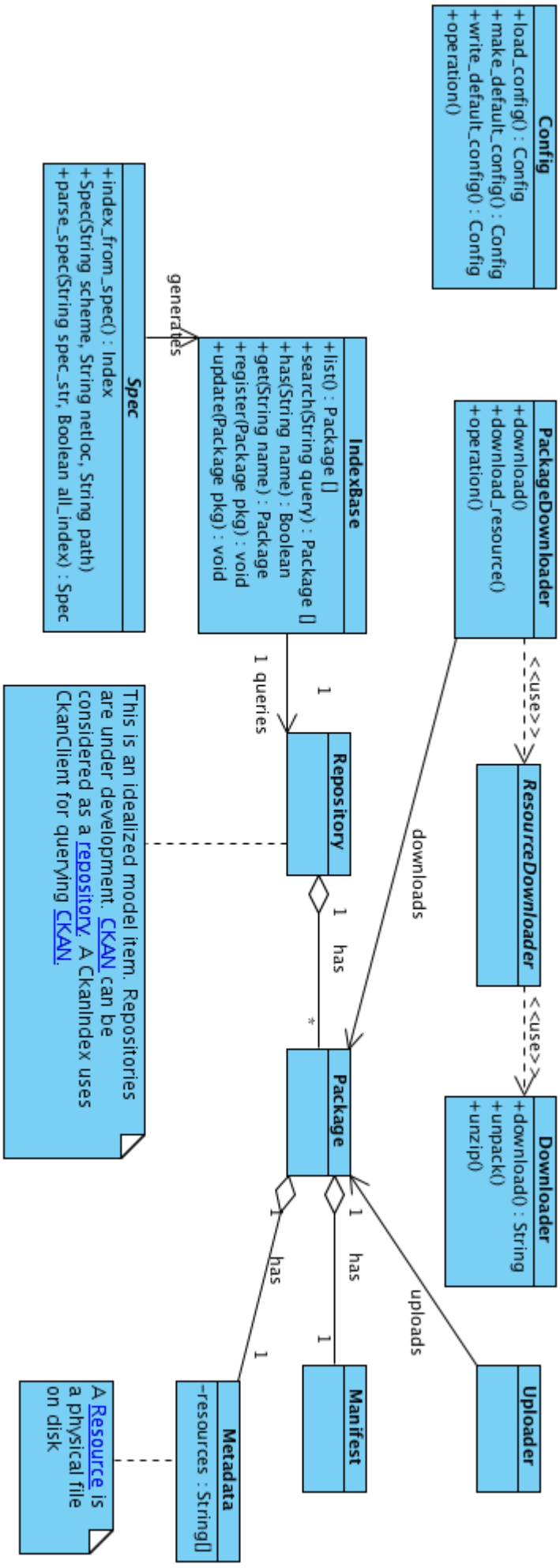

Illustration 2: Datapkg Package

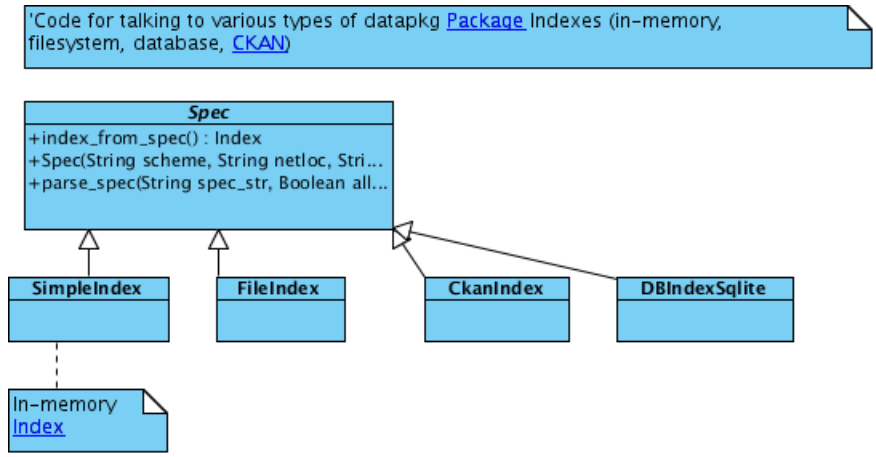

Illustration 3: Index Package

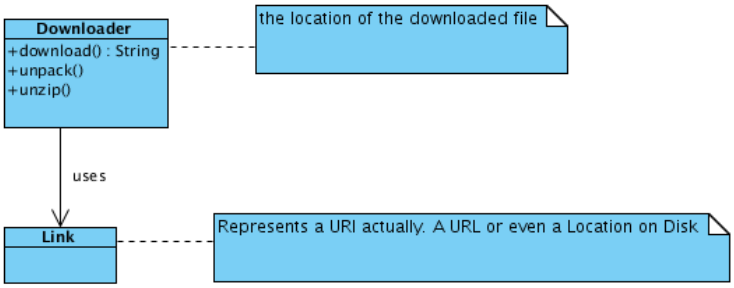

Illustration 4: Util Package

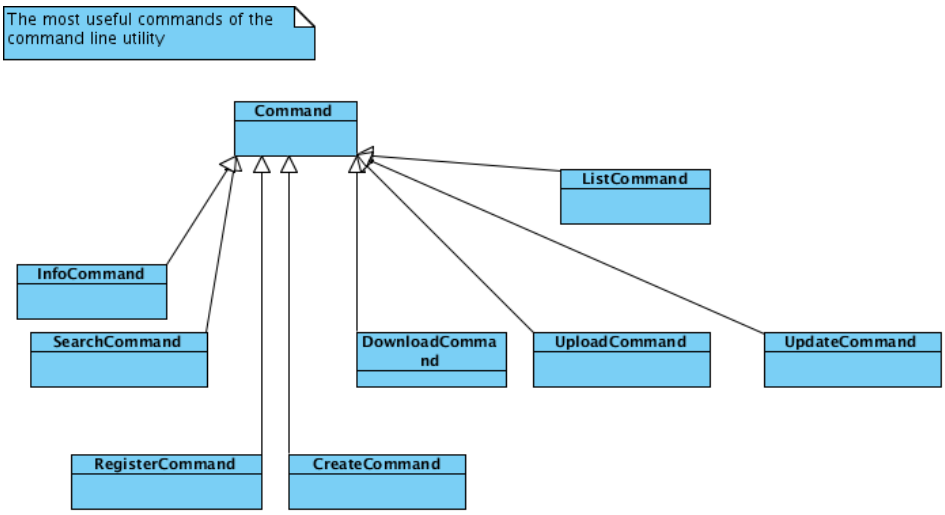

Illustration 5: Command Line Package

Datapkg Components in Action

A better understanding of datapkg components helped us to focus in the domain of interest and learn about nouns and verbs related to the operations to be performed on The Data Hub. To better understand how the components interact with each other, we decided to draw the sequence diagrams of two common datapkg functionalities: search and retrieval of data.

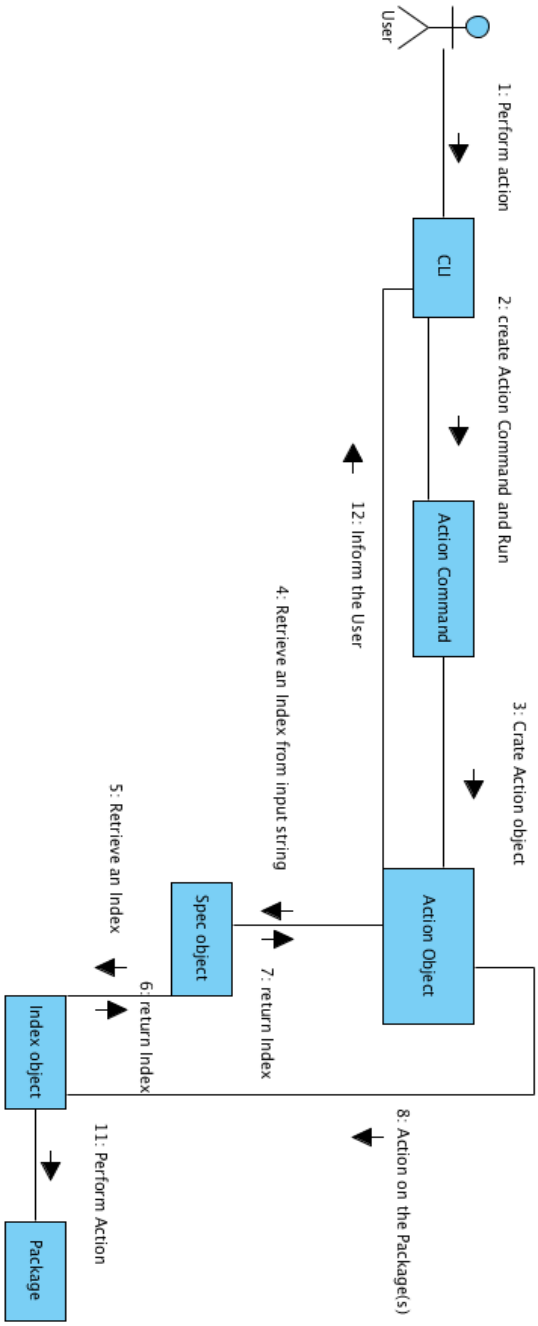

Illustration 6: Generic Pattern of Datapkg

As we can see in Illustration 7 and in Illustration 8, the user interacts with the Command Line that creates Command objects. The user can not go over the Command Line layer using the Command Line. Unfortunately, these objects are designed for the CLI and can not be re-used as an interaction layer for the GUI. Anyway, we were able to draw a generalization of the majority of the operations done by datapkg, that we can see in Illustration 6. Another issue appears: most output is not performed ad the CLI level but one step below. It is not the action Command object that prints information, but the action object.

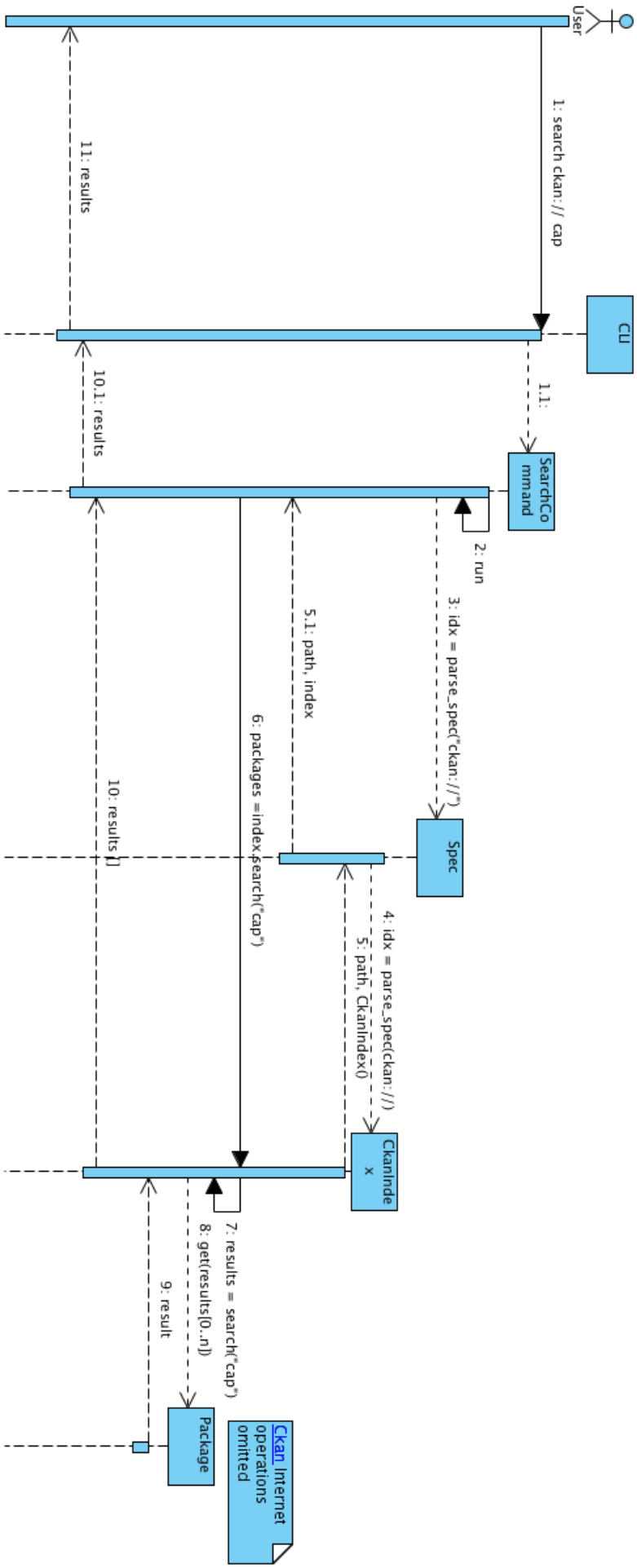

Illustration 7: Search a Package

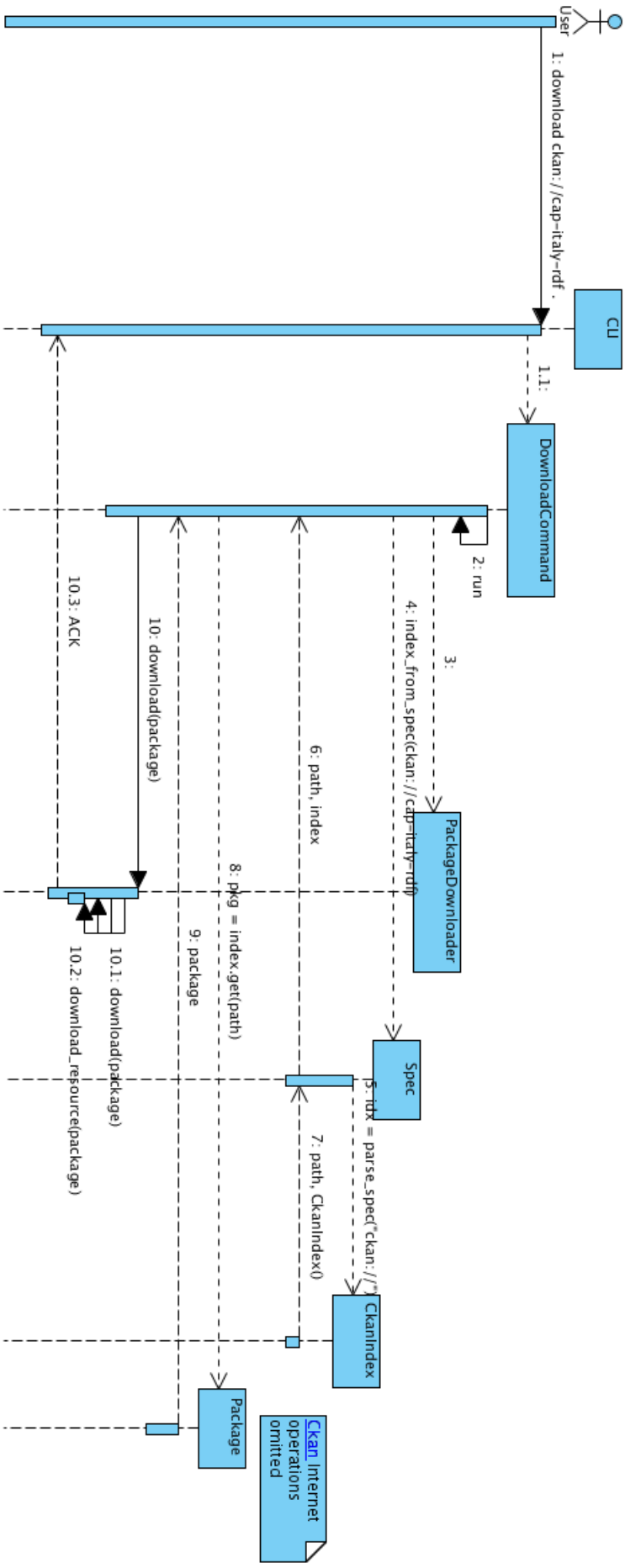

Illustration 8: Download a Package

Datapkg-gui

Our Intervention

After reasoning about the general design/architecture information gathered during the initial phase, we decided that the next step will be to implement a tiny library wrapper for datapkg that behaves exactly like the Command objects declared in the Command Line package. The main difference will be that those library wrappers will be responsible to return meaningful values that the GUI (maybe one day also the Command Line) will interpret in order to return information to the user.
Illustration 9 presents a proposed architecture for datapkg project. It is a simple layered system. On the bottom side we have the Core part, currently coded in datapkg, datapkg.index and datapkg.distribution packages. The Library is the first part of our project and will be coded in a very similar way as the CLI does. On top of the library we have the user interfaces, that are the current CLI (datapkg.cli) and the GUI we will develop. Of course, such layered organization providing an internal API would encourage developers to implement other projects on top of it.
While we don't claim that datapkg must follow our proposal, we strongly encourage this approach.

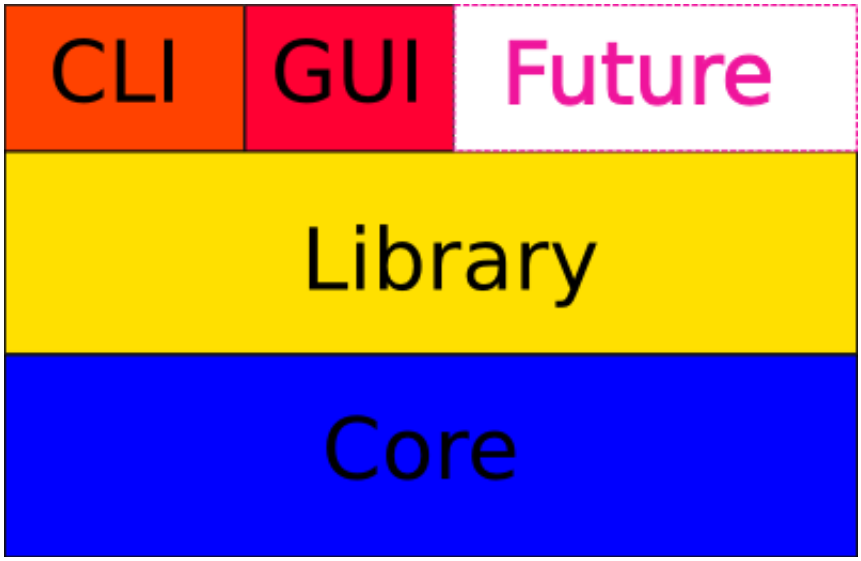

Illustration 9: Proposed Architecture