# Exploring a Data Flow Design of the CARTA System

Technical Documentation

Zainab Adjiet
Department of Computer Science
University of Cape Town
adjzai001@myuct.ac.za

October 5, 2020

# Contents

## List of Figures

## 1 INTRODUCTION

The Cube Analysis and Rendering Tool for Astronomy (CARTA) [3] is designed to visualise and analyse data from the Atacama Large Millimetre Array (ALMA) [1], the Very Large Array [11], and the Square Kilometre Array (SKA) [7] pathfinders. CARTA uses a client-server architecture to visualise the large images obtained from these modern telescopes. The data storage and computation are handled by enterprise-class servers or clusters, and the processed information is sent to the front-end web client for visualisation (see Fig. 1).

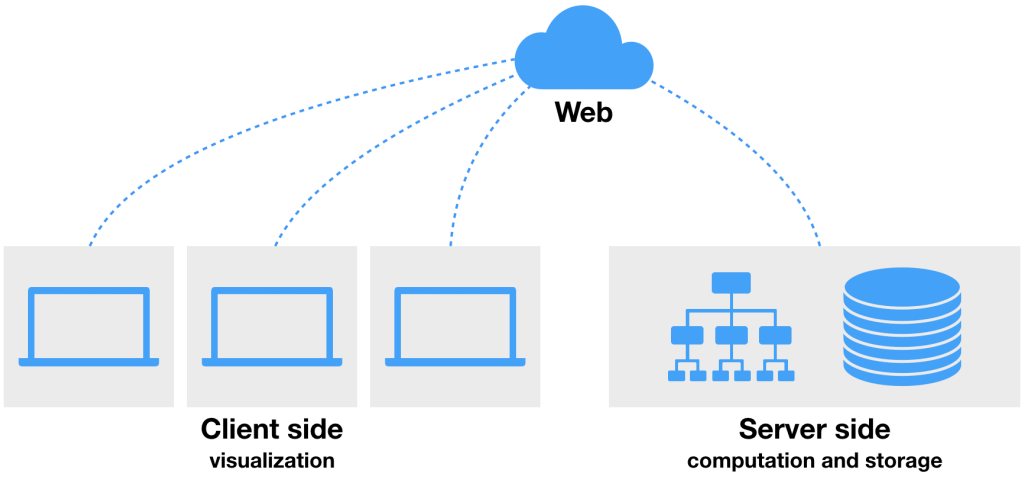


Fig. 1. CARTA Client-Server Architecture [2]

With the first Exaflop computer expected around 2020 [9], the NRAO is exploring the use of a data flow model [6] for the CASA library. The prototype will be implemented in an interpreted language, Python, with the use of the Python Dask library.

Inspired by the work being done by the NRAO, an architectural re-design of the current of the CARTA back-end system was investigated using the Python-based Dask data flow environment to explore the implications of this shift to a data flow model. Alongside this re-design, a prototype was implemented by Dylan Fouche [10], which was ofttimes used as inspiration for the design.

All diagrams used in this investigation were constructed using the free, online Diagrams.net tool [8]. The back-end system use cases were represented in use case diagrams (App. A) which were used to construct structural (App. B) and behavioural (App. C) UML diagrams to represent the proposed system design.

## 2 METHODS

### 2.1 Requirements Capturing

A set of typical use case diagrams was constructed with a CARTA client user as the focus actor of the diagram (Figs. 2, 3, 4, 5) to obtain system requirements from the user's view. These diagrams were based on the sequence diagrams contained in the CARTA ICD [4].

A second set of use case diagrams was created using the CARTA front end as the focus actor (Fig. 6, 7, 8). This second diagram allowed a more in-depth look into the needed functionality of the CARTA back-end system.

### 2.2 Static System Design

A package diagram, paying specific attention to modules used, was constructed (Fig. 9) to depict how the back end should interact with the existing front-end packages and what other packages would be necessary for its functions. The diagram is based on Dylan Fouche's Dask prototype of the CARTA back end [10] and depicts the Python modules that the back end needs to calculate a per-cube histogram.

A standard UML design class diagram was used to narrow down on the classes and functionality within the back end (Fig. 10). Fouche's prototype was used again as a base for the relationships between the classes and some initial class functionality.

Both the package and class diagrams were then revised and restructured (Figs. 11, 12) to take the full back-end system functionality into account. The CARTA Github repository [5] was used as a reference of how the back end can be structured to support the full functionality and inspiration was drawn from this to restructure the design.

### 2.3 Dynamic System Design

Data flow diagrams show how data moves between entities and processes in a system, and this can give the best representation of the Dask data flow environment. A zero-level data flow diagram, commonly referred to as a context diagram, was constructed for the histogramming function of the CARTA system (Fig. 13). Still focusing on this operation, another two data flow diagrams were constructed each with varying levels or depths.

The first-level diagram depicts the movement of data with the histogramming operation being run on a single computing node (Fig. 15). The second-level uncovers the Dask scheduler entity embedded in the back end that orchestrates the operations on a set of Dask worker nodes (Fig. 16). Based on the second-level histogramming data flow diagram, a generalised second-level diagram was created (Fig. 14). This diagram focuses on the interaction of the Dask scheduler with individual worker nodes and shows how the flow can be generalised.

A sequence diagram was constructed (Fig. 17) to depict the flow of the histogramming operation as it would execute on the revised system structure (Fig. 12). This sequence diagram expands on how the back-end classes interact with each other to compare this to how the existing CARTA back end [5] executes the same operation.

## REFERENCES


[1] A Worldwide Collaboration ALMA. 2020. Atacama Large Millimeter/submillimeter Array. https://www.almaobservatory.org/en/home/

[2] CARTA. 2018. CARTA Client-Server Architecture. https://carta.readthedocs.io/en/latest/_static/carta_intro_serverClient.png

[3] CARTA. 2018. *Cube Analysis and Rendering Tool for Astronomy.* https://carta.readthedocs.io/en/latest/index.html

[4] CARTA. 2020. *CARTA Interface Control Document — CARTA Interface Control Document documentation.* https://carta-protobuf.readthedocs.io/en/latest/index.html#

[5] CARTAvis community. 2020. *CARTA Backend.* https://github.com/CARTAvis/carta-backend/

[6] David E Culler. 1986. Dataflow architectures. *Annual review of computer science* 1, 1 (1986), 225–253.

[7] David Davidson. 2012. MeerKAT and SKA phase 1. *2012 10th International Symposium on Antennas, Propagation and EM Theory, ISAPE 2012*, 1279–1282. https://doi.org/10.1109/ISAPE.2012.6409014

[8] diagrams.net. 2005. Diagram Software and Flowchart Maker. https://www.diagrams.net/

[9] Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, et al. 2011. The international exascale software project roadmap. *The international journal of high performance computing applications* 25, 1 (2011), 3–60.

[10] Dylan Fouche. 2020. *CADaFloP.* https://github.com/DylanFouche/CADaFloP/

[11] National Radio Astronomy Observatory. 2020. *Very Large Array - National Radio Astronomy Observatory.* https://public.nrao.edu/telescopes/vla/

## A REQUIREMENTS CAPTURING DIAGRAMS

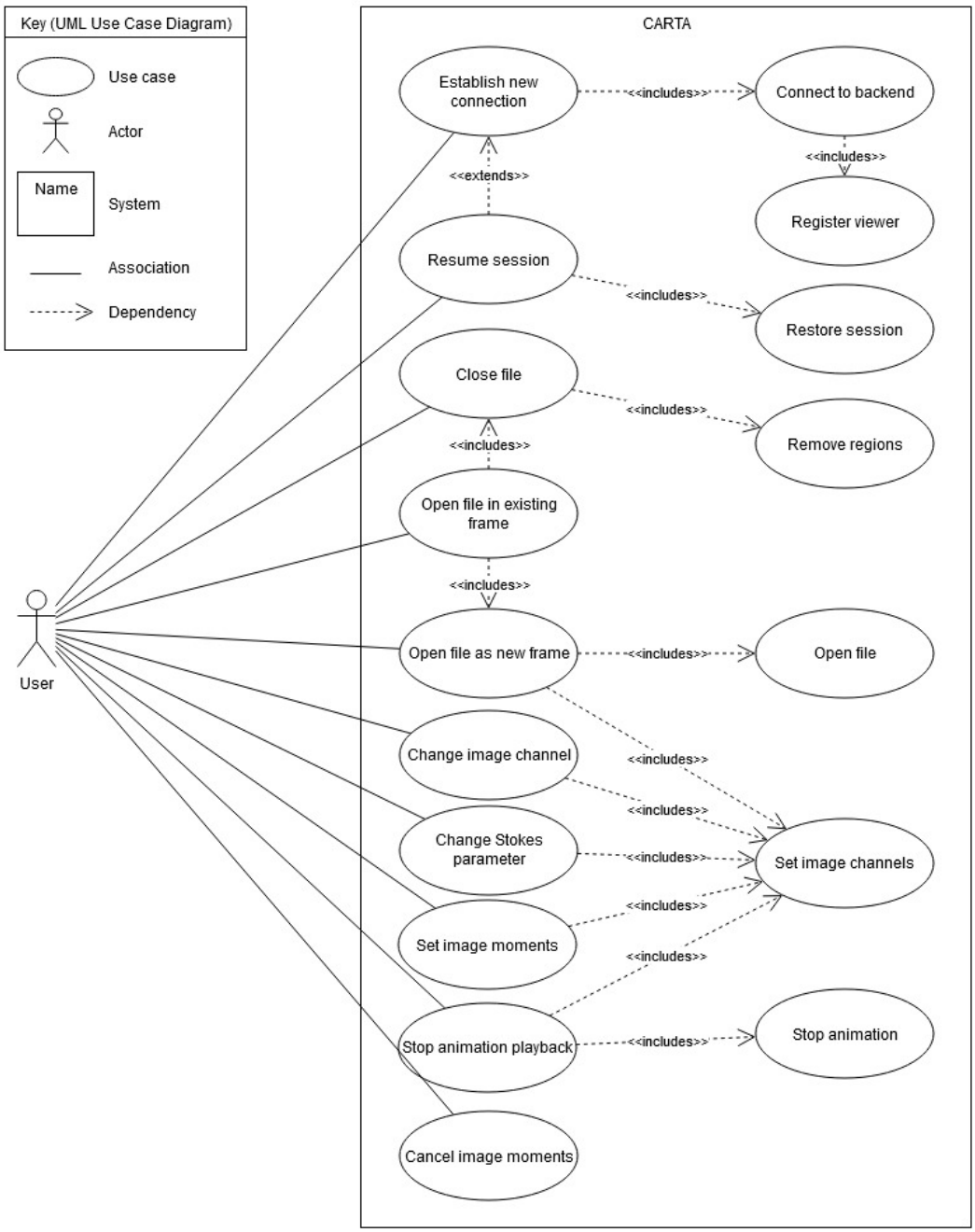


Fig. 2. Part 1 of CARTA Use Case Diagram Set

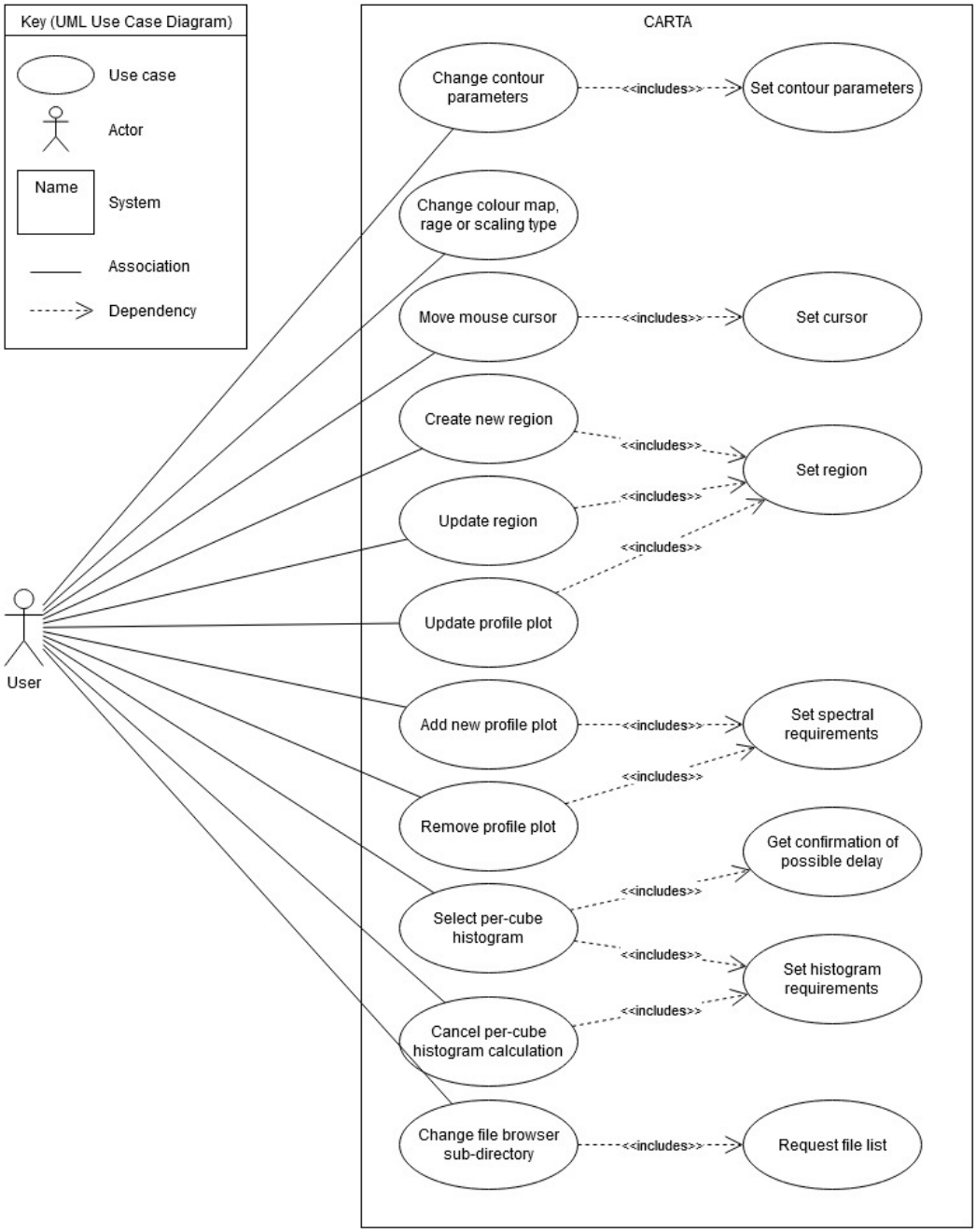


Fig. 3. Part 2 of CARTA Use Case Diagram Set

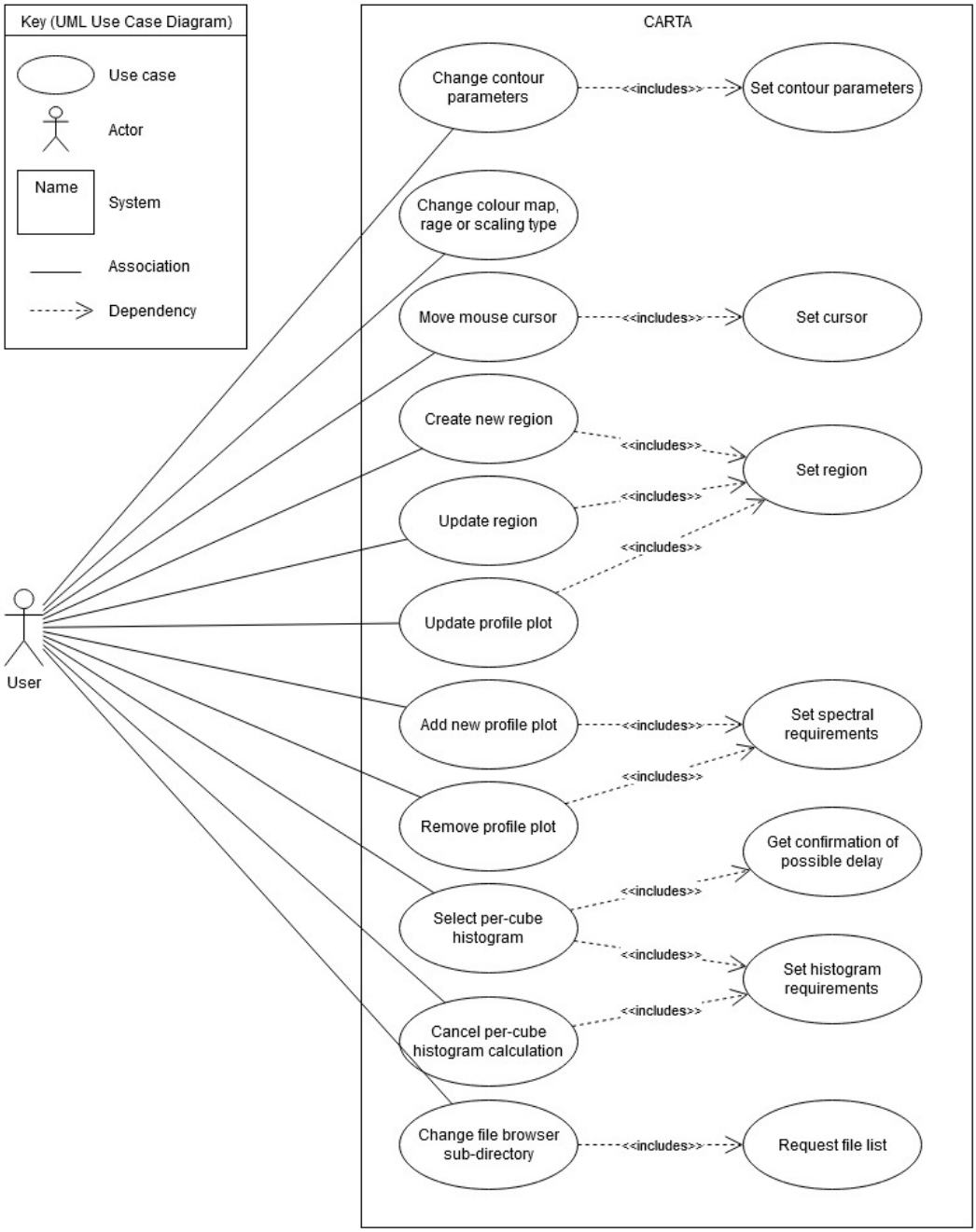


Fig. 4. Part 3 of CARTA Use Case Diagram Set

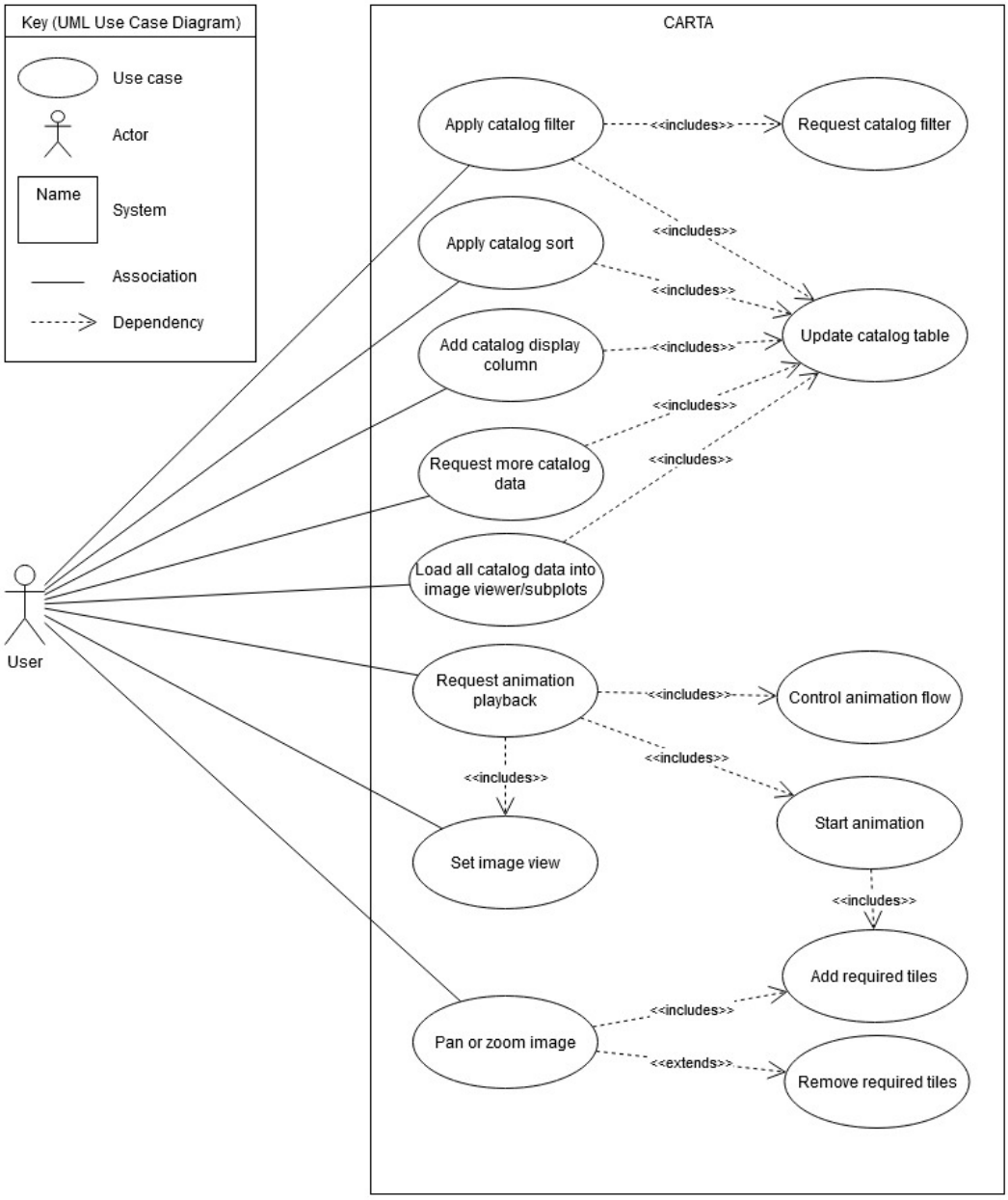


Fig. 5. Part 4 of CARTA Use Case Diagram Set

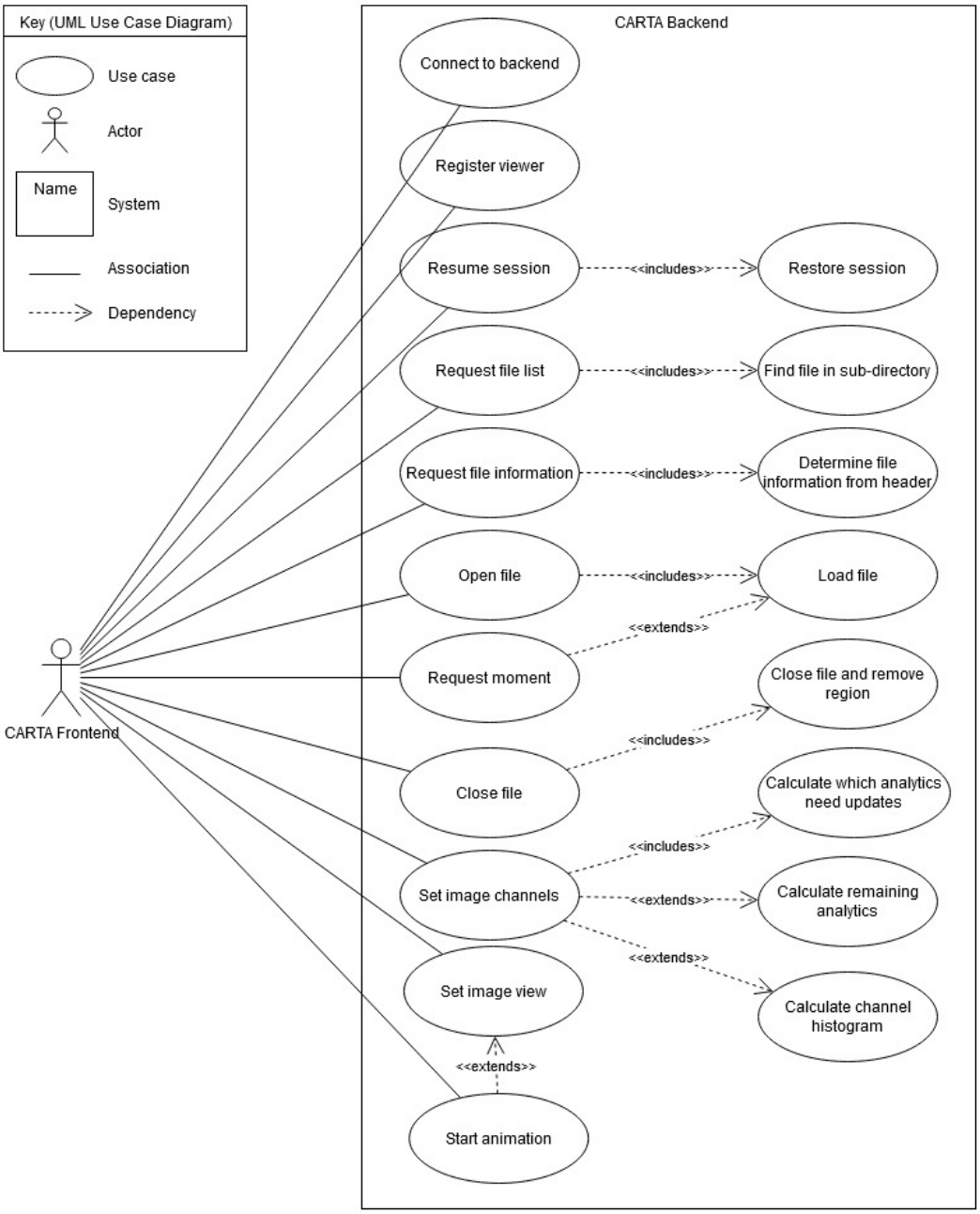


Fig. 6. Part 1 of CARTA Front End Use Case Diagram Set

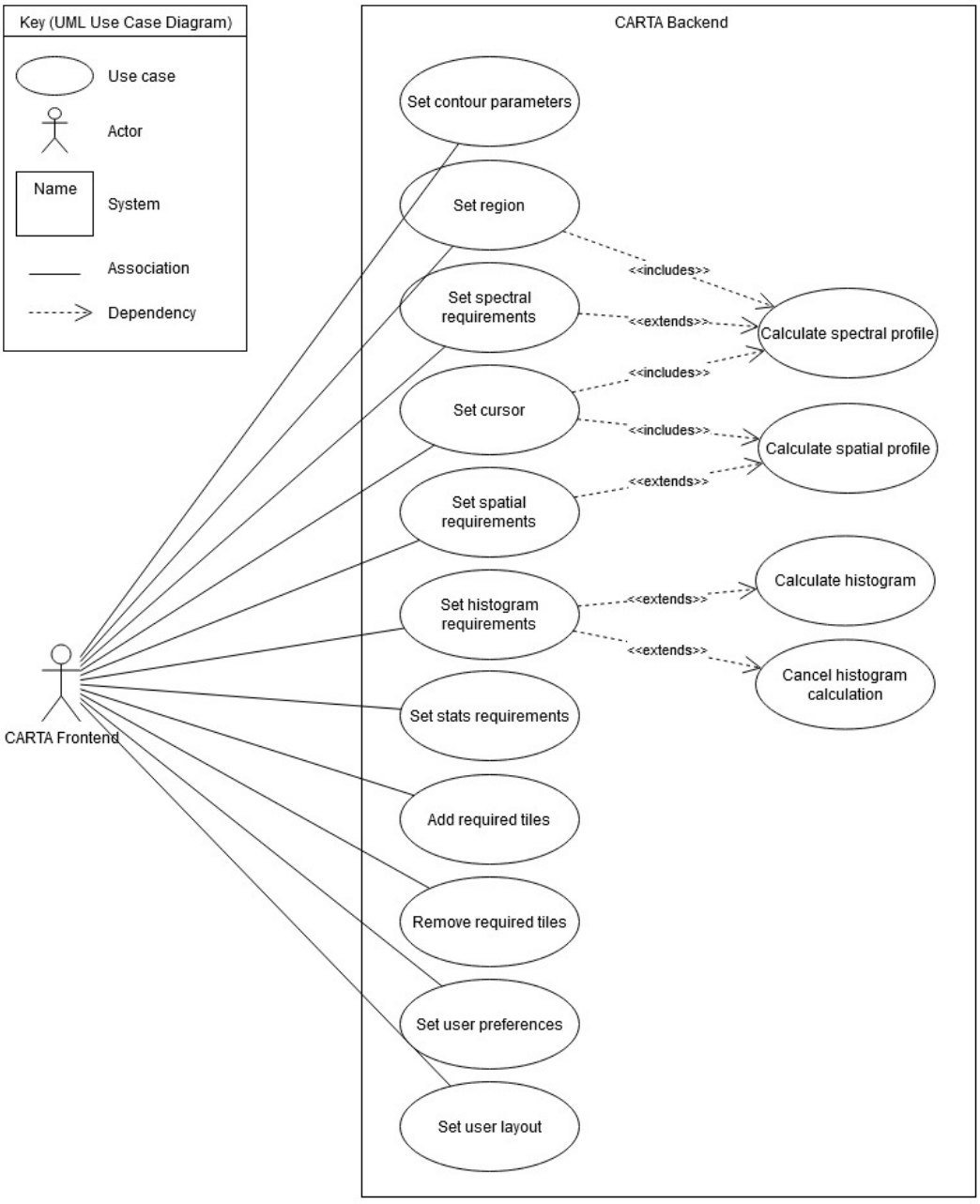


Fig. 7. Part 2 of CARTA Front End Use Case Diagram Set

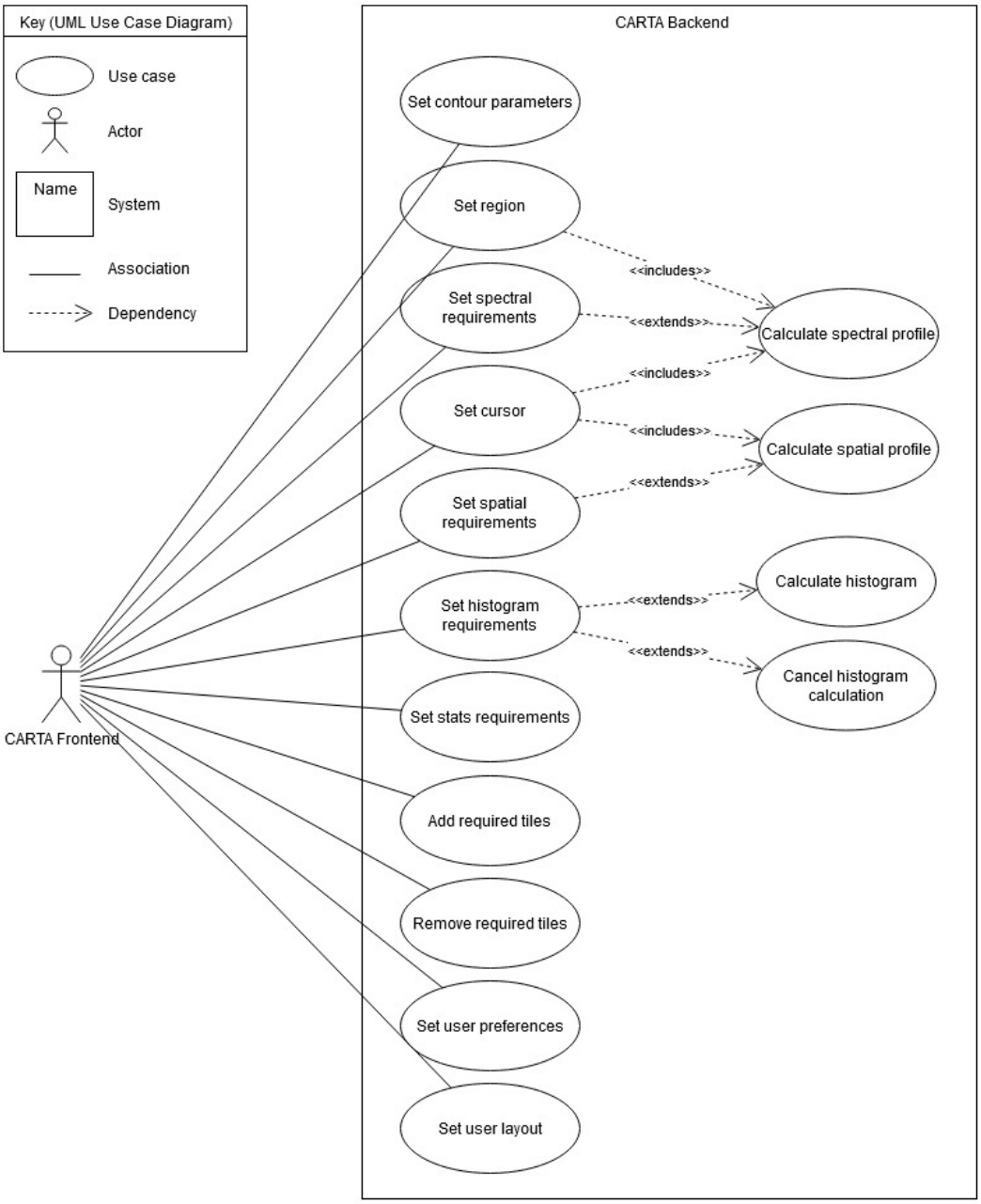


Fig. 8. Part 3 of CARTA Front End Use Case Diagram Set

## B STATIC SYSTEM DESIGN DIAGRAMS

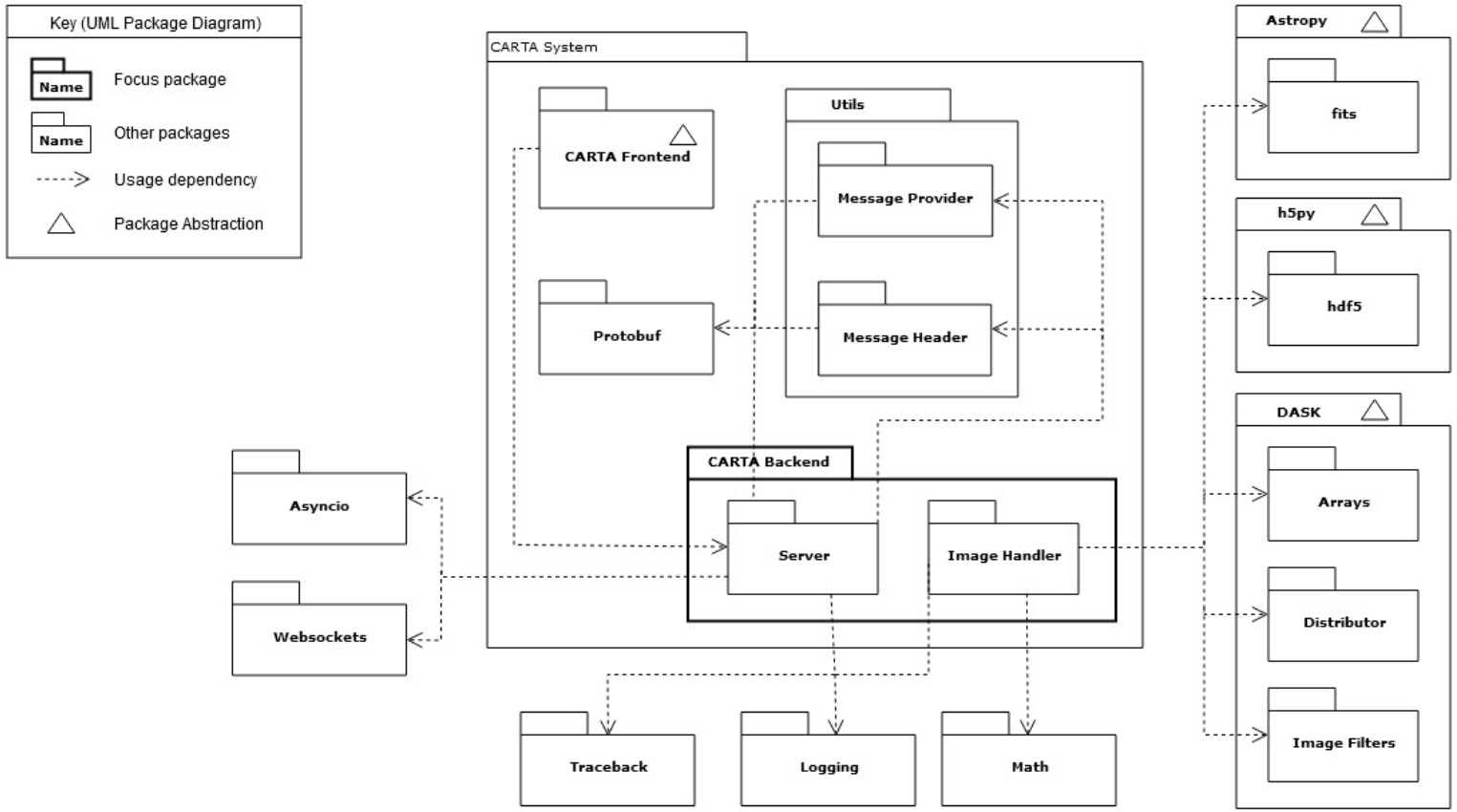


Fig. 9. CARTA Back End Package Diagram based on Fouche's Prototype [10]

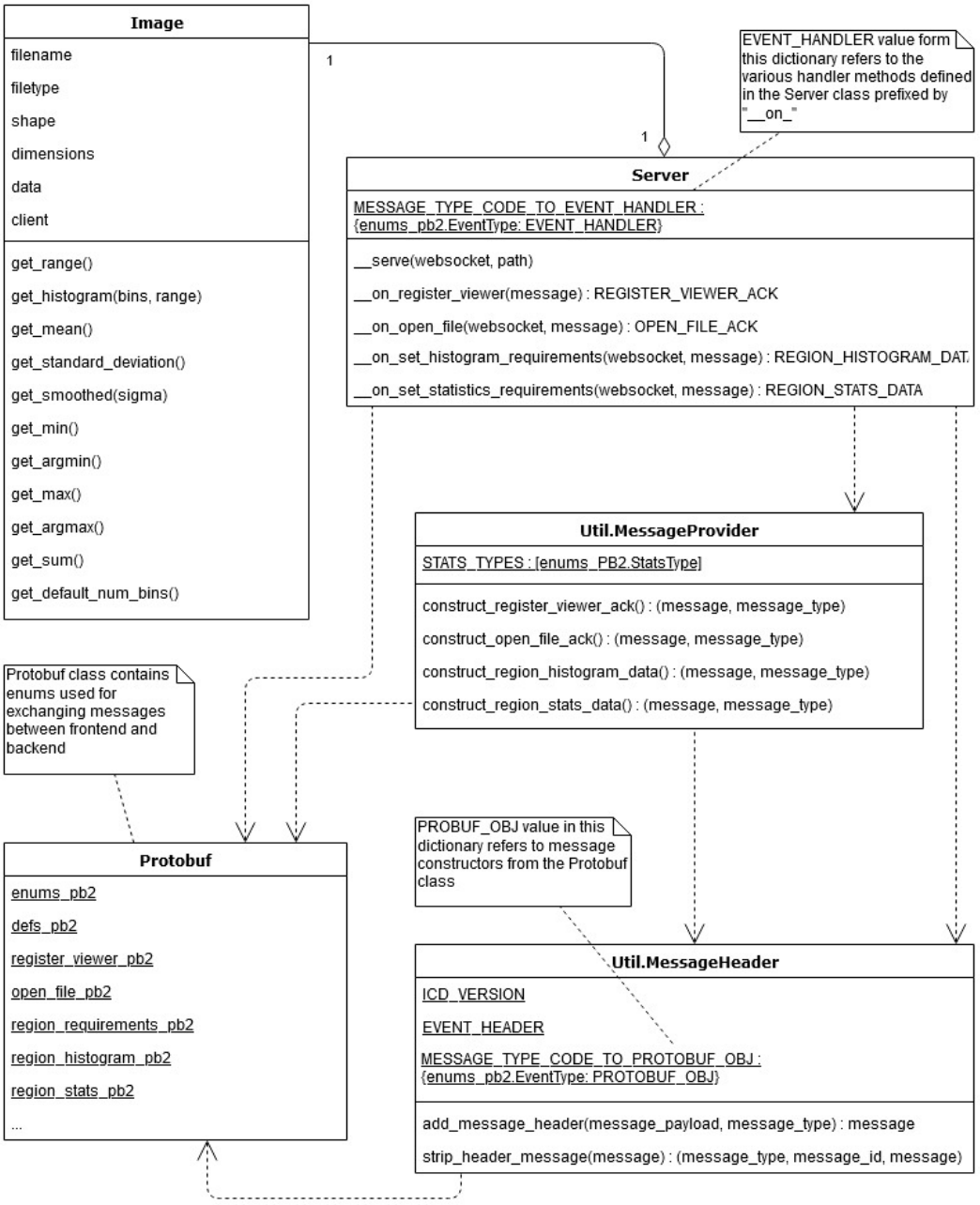


Fig. 10. CARTA Back End Class Diagram based on Fouche's Prototype [10]

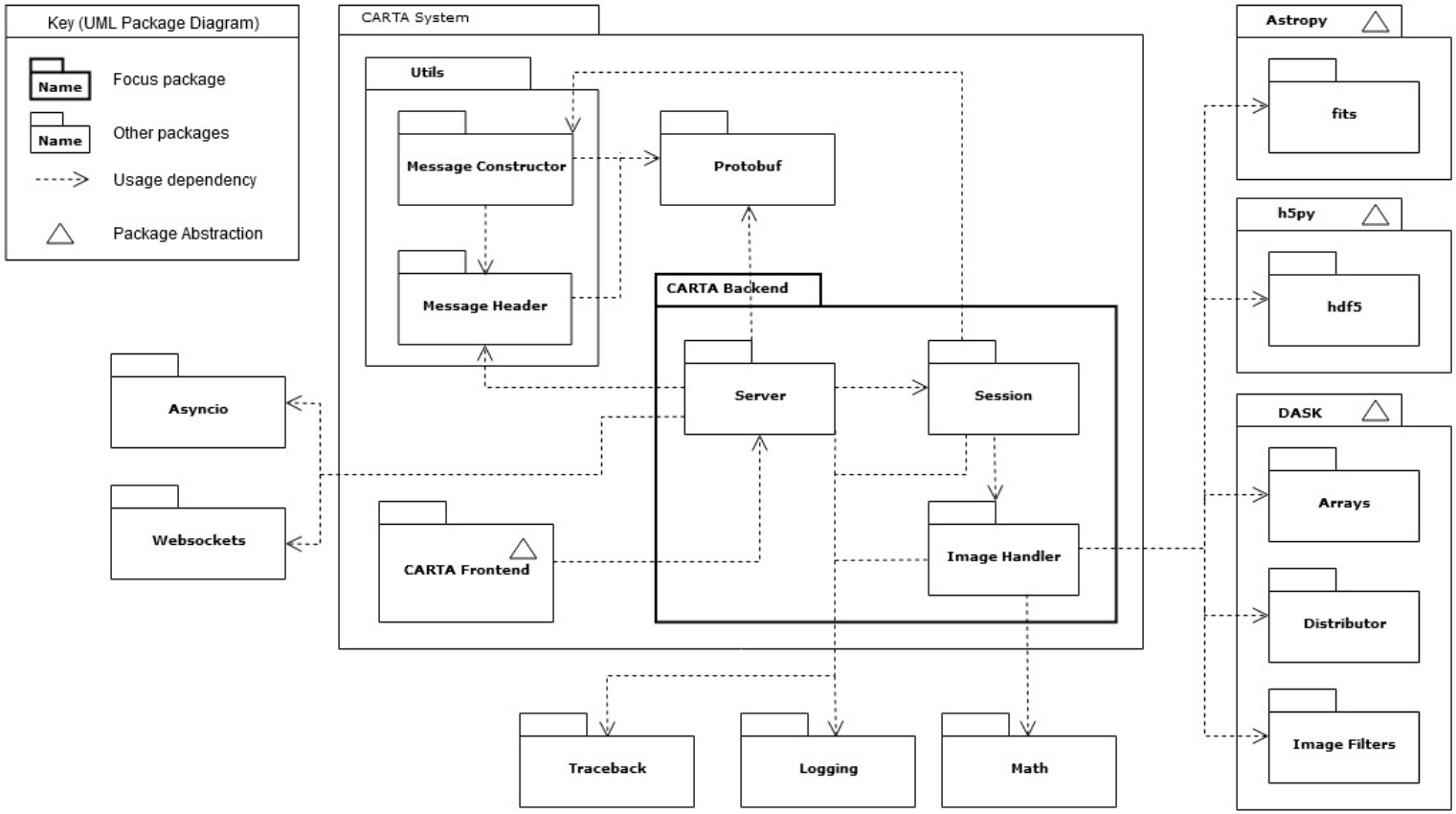


Fig. 11. Revised Full-Functionality CARTA Back End Package Diagram

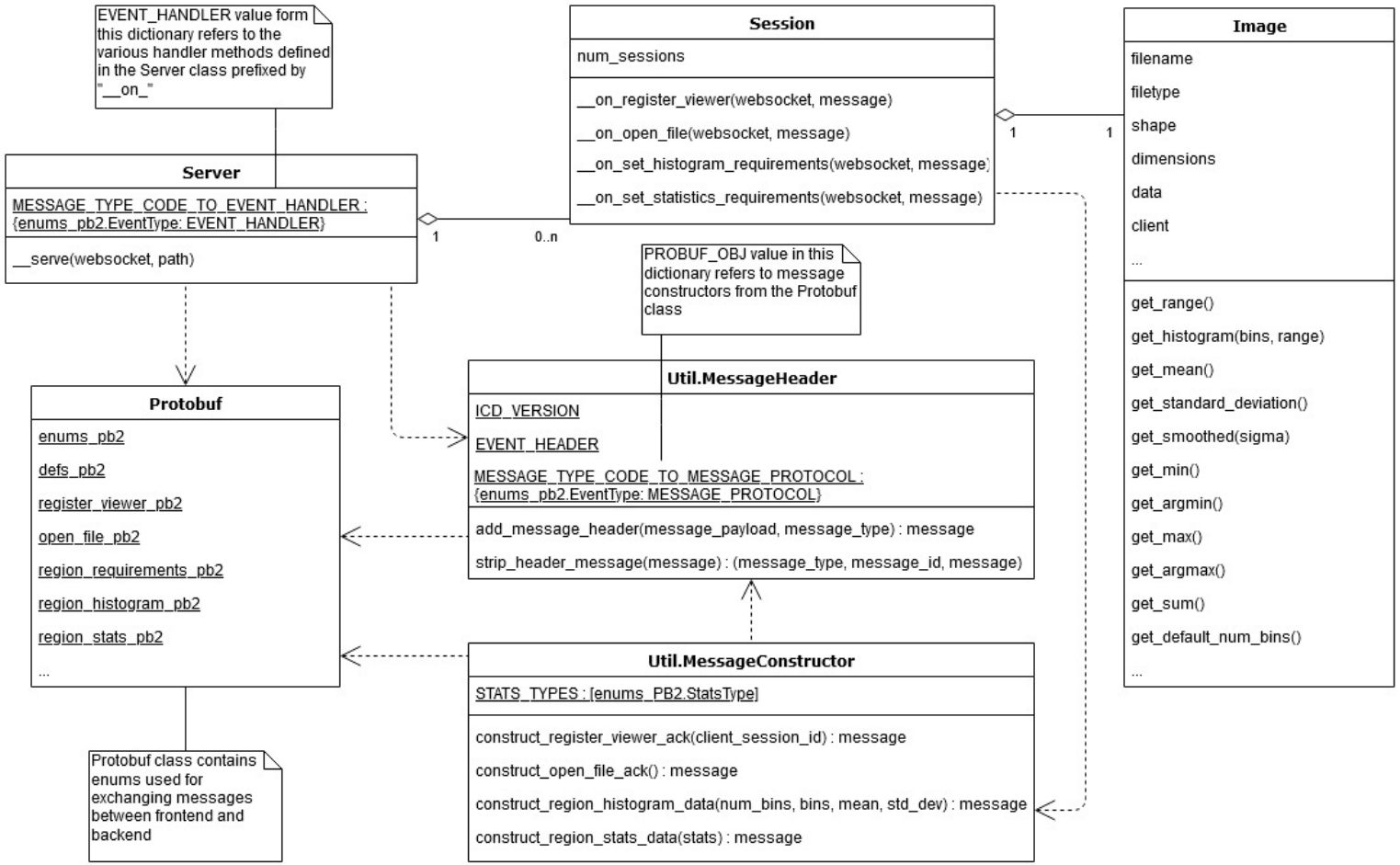


Fig. 12. Revised Full-Functionality CARTA Back End Class Diagram

## C DYNAMIC SYSTEM DESIGN DIAGRAMS

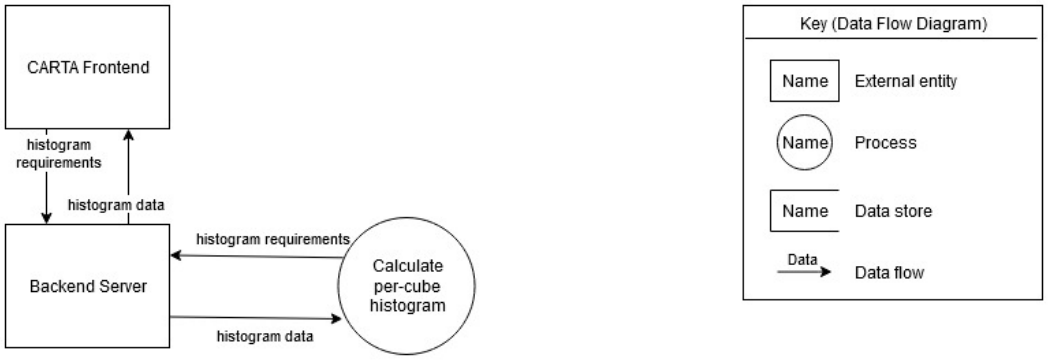


Fig. 13. CARTA Per-Cube Histogram Calculation Context Diagram

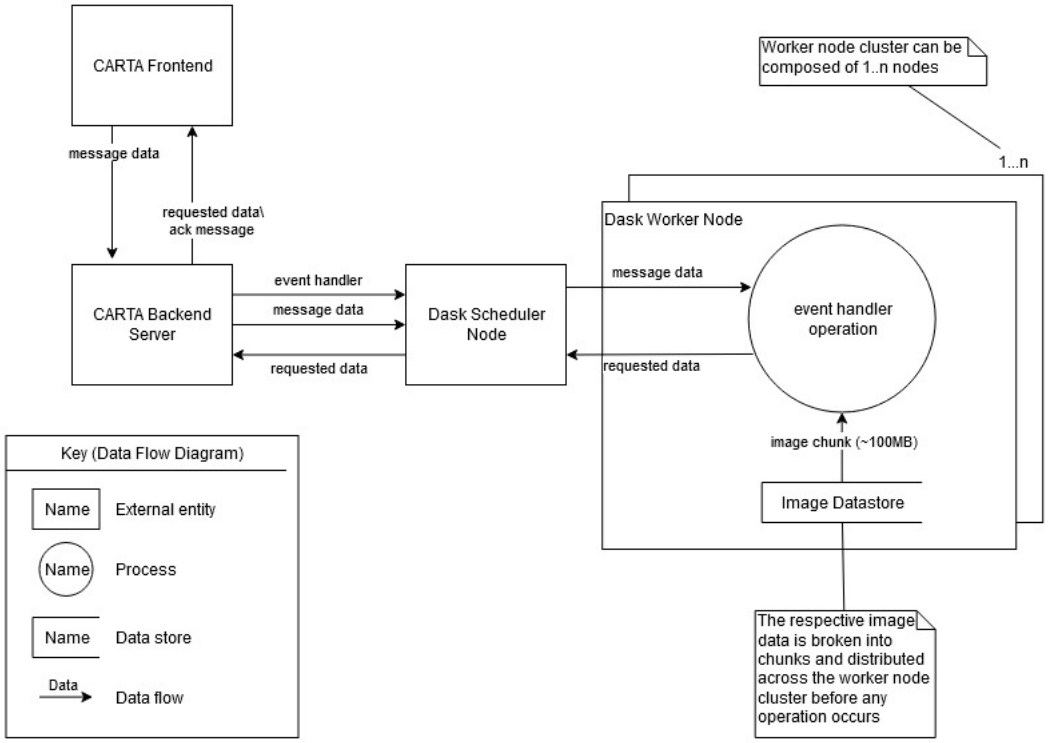


Fig. 14. CARTA Generalised Data Flow Diagram (Level 2)

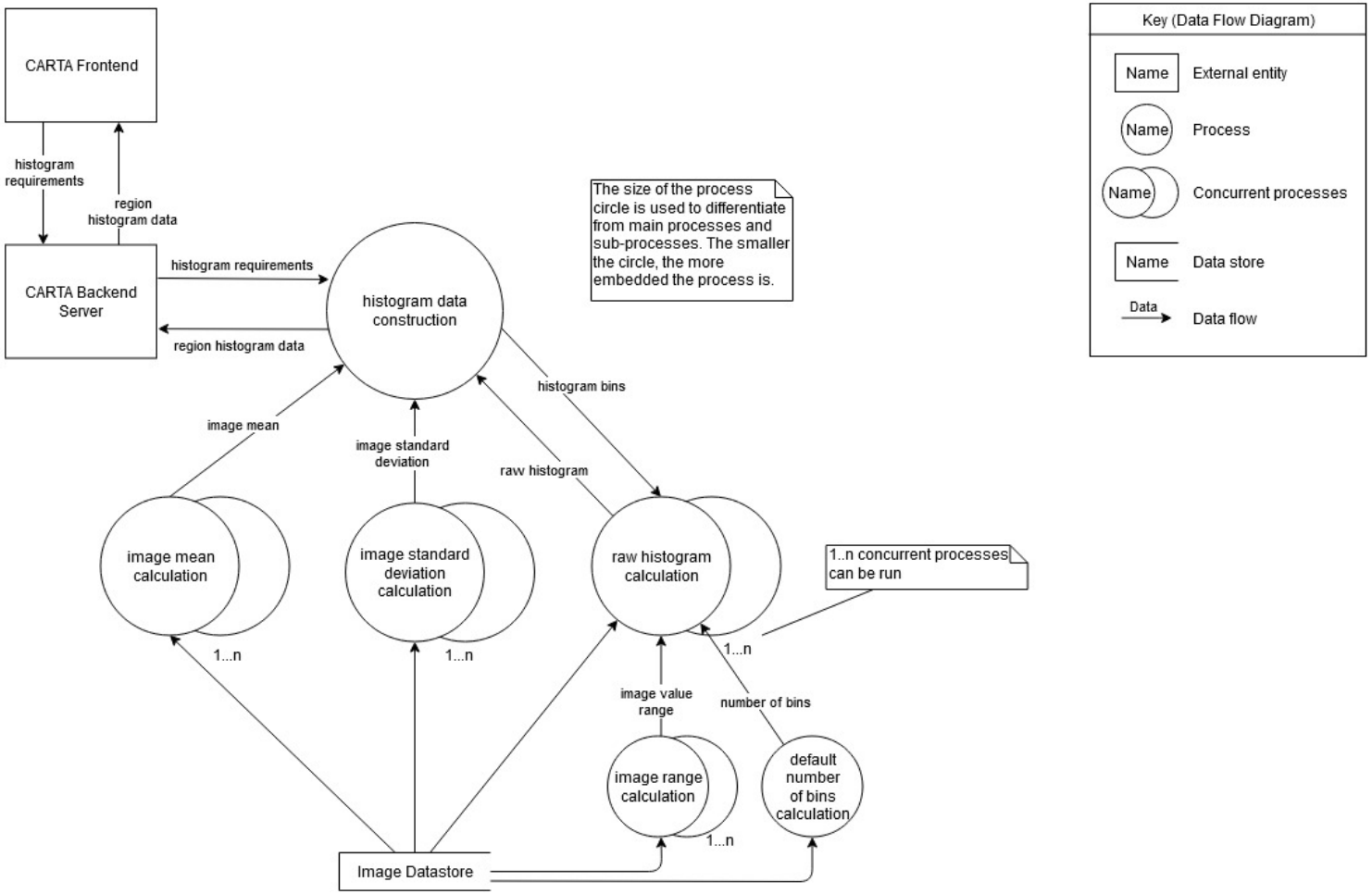


Fig. 15. CARTA Per-Cube Histogram Calculation Data Flow Diagram (Level 1)

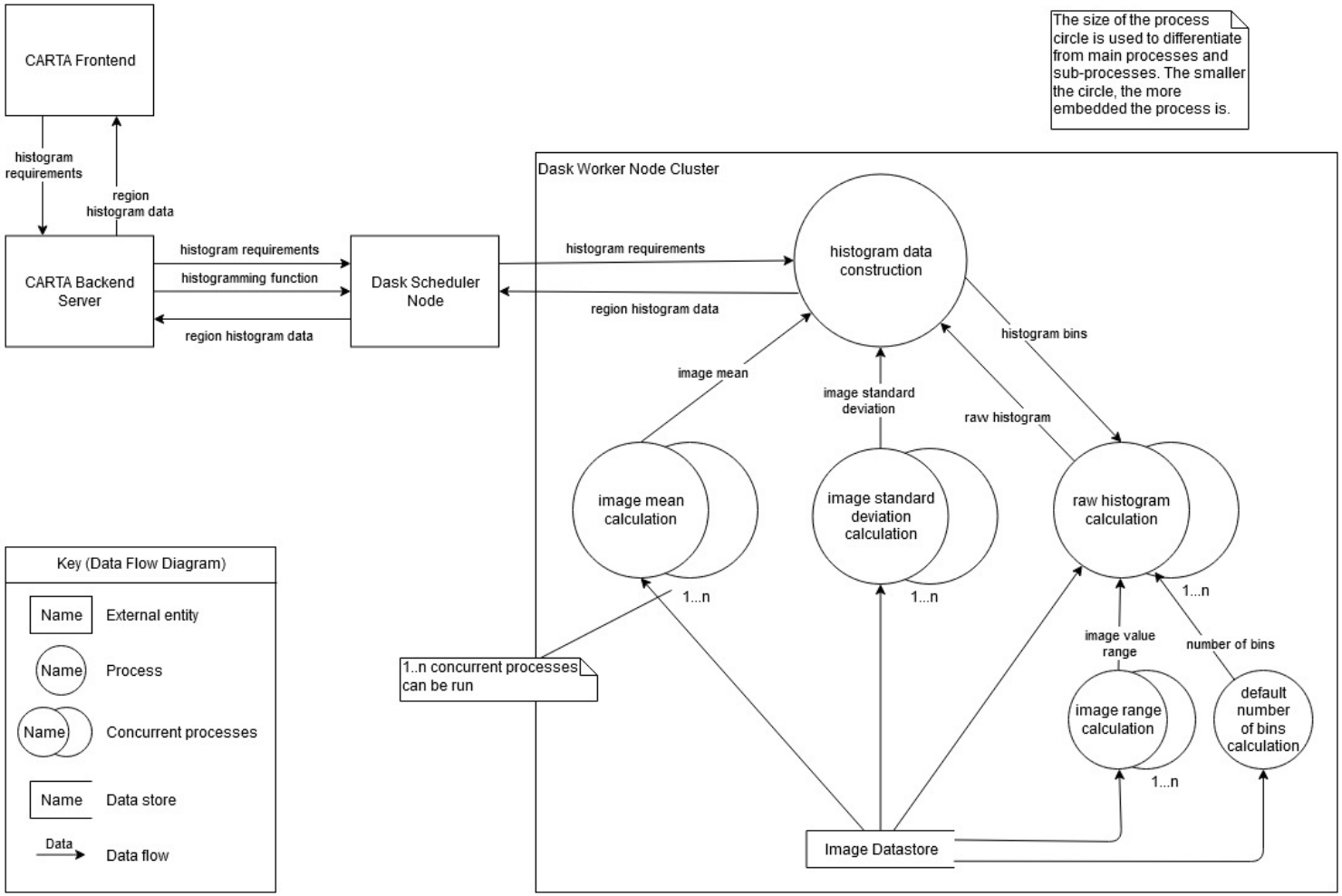


Fig. 16. CARTA Per-Cube Histogram Calculation Data Flow Diagram (Level 2)

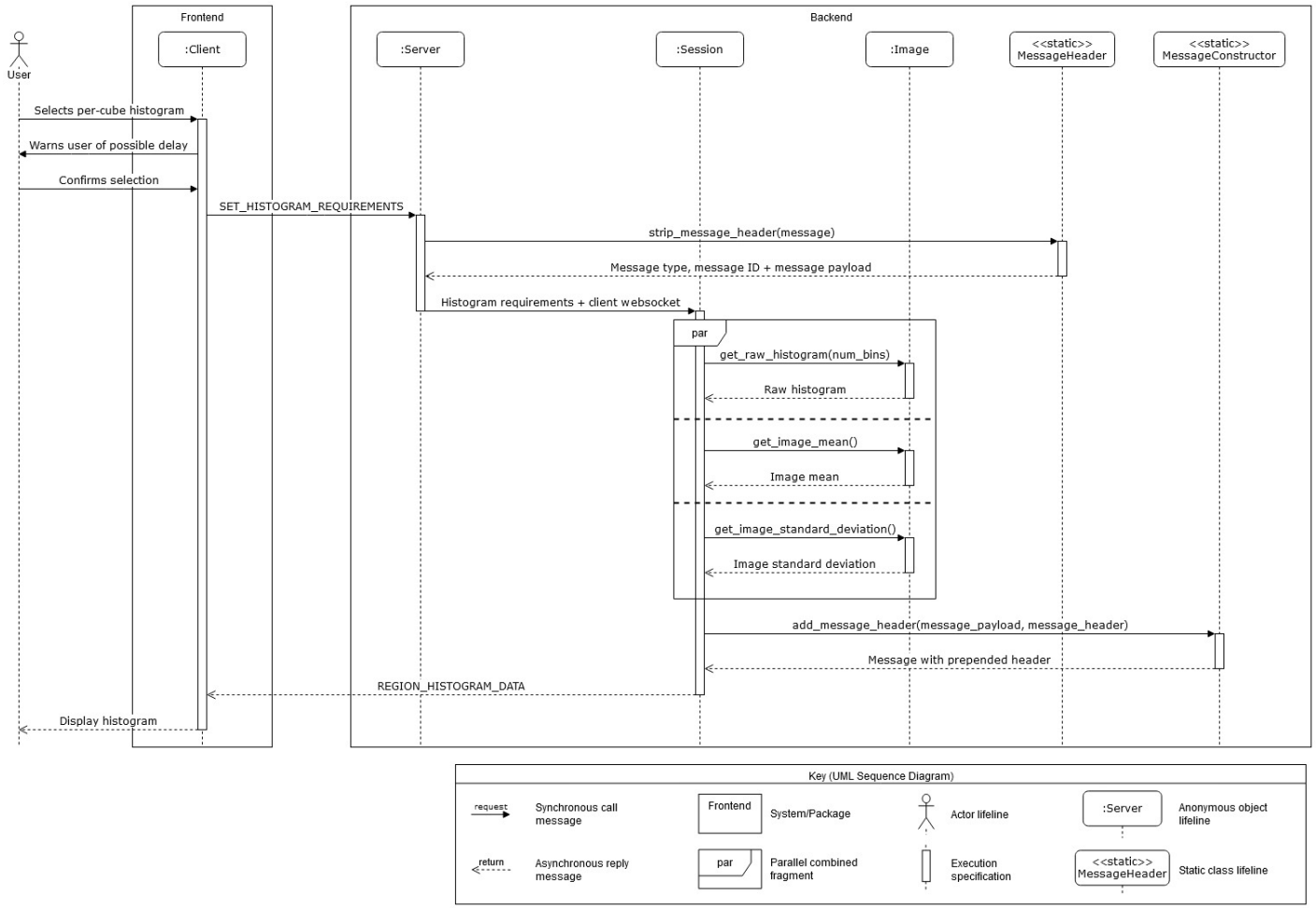


Fig. 17. CARTA Per-Cube Histogram Calculation Sequence Diagram