**CHAPTER 3**

**IMPLEMENTATION**

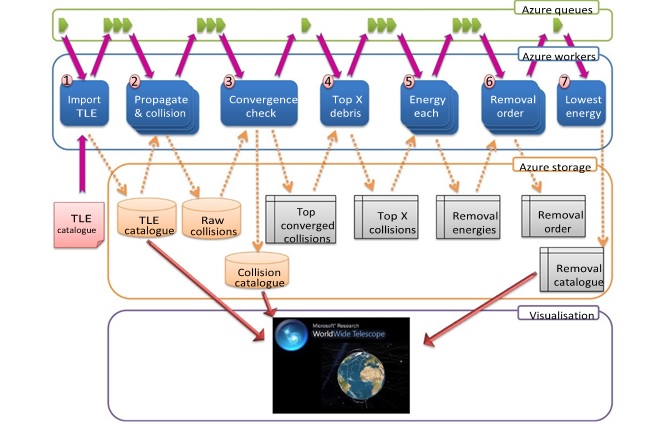
**3.1 A cloud-based architecture solution to the SSA**

Recent computer modelling studies have suggested that the LEO debris population may be stabilised at current levels through the removal of five large, intact objects per year. Whilst this approach can only be successful if the objects that are targeted would otherwise contribute to future collision activity, it does provide a more cost-effective approach to remediation than the removal of all debris objects. However, this leads to a requirement that future collisions are forecast to a sufficient accuracy. In addition, to limit the generation of more debris and to reduce costs further, it is likely that an ADR mission will aim to remove more than one debris object. Consequently, mission requirements include orbital transfers between targets in addition to manoeuvres in close proximity to these uncontrolled objects.

**3.2 Active Debris Removal architecture**

In the light of these requirements, a key concern in the design of an ADR mission will arise from the choice of propulsion system. The choice will be determined, in part, by the energy required to remove debris targets from orbit and to transfer to subsequent targets. The required energy also provides an additional constraint on the selection of removal targets, as it is also linked to mission cost, such that the determination of the route between target destinations becomes an important optimisation task in ADR mission design. This optimisation problem, known as the travelling purchaser problem (TPP), forms the basis of the demonstration of a cloud-based computing approach. The cloud-based architecture for the example ADR mission. The architecture is implemented on Microsoft Windows Azure and each numbered block in the figure is a worker type which can be launched as multiple instances if required.

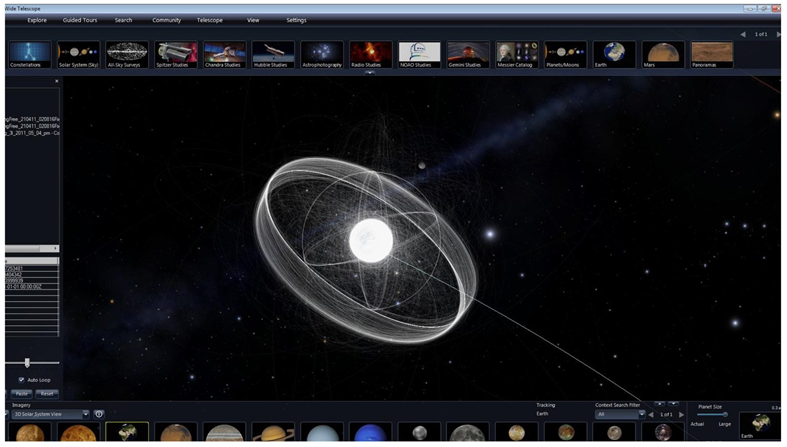
ADR mission with a chemical propulsion system performs a rendezvous manoeuvre to attach a solid rocket motor to a target object, which subsequently fires under remote command to de-orbit the target. The ADR vehicle then uses its primary chemical propulsion to transfer to the next target. Removal targets are identified and ranked using a fast, pair-wise collision algorithm based on the Cube approach employed by the LEO-to-GEO Environment Debris Model (LEGEND) and applied to all objects in the US SSN catalogue. The approach determines the collision probability for each object using Monte Carlo (MC) simulation, whereby the number of MC samples effectively determines the amount of compute time required within the cloud-based solution. The entire workflow is a single instance of the ADR analysis for a given point in time and it is possible to run multiple workflows in parallel; they do not require inter-process communications. In this paper we only run a single ADR instance, which is comprised of seven different worker types.



***Fig 3.1:*** *Windows Azure active debris removal architecture*

The ADR architecture generates data which is stored in cloud-storage. Accessing the raw data from cloud-storage is trivial and we utilise World Wide Telescope (WWT) to visualise the input, output and intermediate files. WWT has a rich API which supports importing data via a REST interface or from Excel, and is used to visualise data directly from Windows Azure.

The workers depicted with a single block are single instance workers, where as those with multiple blocks are parallel workers. The propagation and collision worker consumes the largest computational resources but is highly parallel, however the convergence checker cannot start until the propagator has completed. Currently, convergence is checked after the propagator has run for a set number of times, but future implementations will run a convergence checker which can terminate the propagation once converged (to save computational resources).



***Fig 3.2:*** *World wide telescope displaying a full TLE catalogue.*

This limits the minimum computational wall clock time, if each propagation and collision MC simulation was run with its own worker, the minimum time to complete a full run is bounded below by the time taken by a single propagation (this could be accelerated by using more powerful hardware). The ‘energy each’ and ‘removal order’ workers have to wait for the top debris list, but can then process the entries in parallel. Using cloud storage and queues reduces the communication bottlenecks and failure overheads as they are transactional and fault-tolerant.