

Prototyping operational autonomy for Space Traffic Management

Sreeja Nag^{a,*}, David D. Murakami^b, Nimesh A. Marker^c, Miles T. Lifson^a,
Parimal H. Kopardekar^b

^a NASA Ames Research Center and Bay Area Environmental Research Institute, CA, USA

^b NASA Ames Research Center, CA, USA

^c NASA Ames Research Center and Stinger Ghaffarian Technologies, Inc., CA, USA

ARTICLE INFO

Keywords:

Space debris
Space traffic
Decision making under uncertainty
Trajectory optimization
Spacecraft management

ABSTRACT

Current state of the art in Space Traffic Management (STM) relies on a handful of providers for surveillance and collision prediction, and manual coordination between operators. Neither is scalable to support the expected 10x increase in active spacecraft population in less than 10 years, nor does it support automated maneuver planning. We present a software prototype of an STM architecture based on open Application Programming Interfaces (APIs), and drawing on insights from NASA's architecture for low-altitude Unmanned Aerial System Traffic Management. The STM architecture is designed to provide structure to the interactions between spacecraft operators, various regulatory bodies, and STM service suppliers, while maintaining flexibility of these interactions and the ability for new market participants to enter easily. Autonomy will be an indispensable part of the proposed architecture in enabling efficient data sharing, coordination between STM participants and safe flight operations (e.g. select spacecraft maneuvers to prevent impending conjunctions between multiple spacecraft).

The STM prototype is based on modern micro-service architecture adhering to OpenAPI standards and deployed in industry-standard virtualized containers, facilitating easy communication between different participants or services. The system architecture is designed to facilitate adding and replacing services with minimal disruption. We have implemented some example participant services (e.g. a space situational awareness/SSA provider, a conjunction assessment supplier/CAS, an automated maneuver advisor/AMA) within the prototype. Different services, with creative algorithms folded into them, can fulfill similar functional roles within the STM architecture by flexibly connecting to it using pre-defined APIs and data models, thereby lowering the barrier to entry of new players in the STM marketplace.

We demonstrate the STM prototype on a multiple conjunction scenario with multiple maneuverable spacecraft, where an example CAS and AMA can recommend optimal maneuvers to the spacecraft operators, based on a pre-defined reward function. Such tools can intelligently search the space of potential collision avoidance maneuvers with varying parameters like lead time and propellant usage, to optimize a customized reward function, and be implemented as a scheduling service within the STM architecture. The case study shows an example of autonomous maneuver planning using the API-based framework. As satellite populations and predicted conjunctions increase, an STM architecture can facilitate seamless information exchange related to collision prediction and mitigation among various service applications on different platforms and servers. The availability of such an STM network also opens up new research topics on satellite maneuver planning, scheduling and negotiation across disjoint entities.

1. Introduction to STM architecture

As outer space becomes increasingly congested with satellites, due to miniaturizing hardware, cheaper launches, more automated operations, entry of emerging economies and proposed large LEO constellations, the active satellite population in Low Earth Orbit (LEO) is expected to grow

from ~1000 to over 16000 in 10–20 years [1–3]. The increased population will be susceptible to greater risk of physical collision with each other or debris, radio-frequency interference, space weather, lasers and directed energy impacts, and thus have the potential to create more debris exponentially (Kessler Syndrome). While large and active satellites are GPS-enabled and propulsion-controlled, small satellites rarely have propulsion and not all have GPS cards. Clusters of small satellites or

* Corresponding author.

E-mail addresses: sreeja.nag@nasa.gov, david.d.murakami@nasa.gov (S. Nag).

<https://doi.org/10.1016/j.actaastro.2020.11.056>

Received 18 May 2020; Received in revised form 22 November 2020; Accepted 28 November 2020

Available online 4 December 2020

0094-5765/© 2020 IAA. Published by Elsevier Ltd. All rights reserved.

Acronyms

AI	Artificial Intelligence	MSMA	Multi-spacecraft Maneuver Advisor Algorithm
AMA	Automated Maneuver Advisor	NORAD	North American Aerospace Defense Command
AGI	Analytical Graphics, Inc.	O/O	Owner/Operator
API	Application Programming Interface	PoC	Probability of Collision
ARC	Ames Research Center	RSO	Resident Space Objects
CAS	Conjunction Assessment Supplier	S3	Space Service Supplier
CCSDS	Consultative Committee for Space Data Systems	SDC	Space Data Center
CDM	Conjunction Data Message	SG	STM Gateway
COLA	Collision Avoidance	SSA	Space Situational Awareness
CSpOC	Combined Space Operations Center	STK	Systems Tool Kit
FAA	Federal Aviation Administration	STM	Space Traffic Management
HIE	High Interest Event	TCA	Time to Closest Approach
IADC	Interagency Space Debris Coordination Committee	TCL	Technology Capability Level
MDP	Markov Decision Process	TLE	Two Line Elements
		UI	User Interface
		UUID	Universally Unique Identifier

massive derelicts thus pose major risks [4]. There are more than 19000 resident space objects (RSOs) greater than roughly 10 cm in size, being currently tracked in Earth orbit. With the U.S. Space Fence expected to be operational soon, better surveillance of RSOs down to 5 cm is planned [5] and even 1–2 cm may be possible [6]. Better resolution and improved orbit determination will elicit more collision avoidance (COLA) maneuvers for previously unavailable tracks and reduce COLA maneuvers for previously false positive tracks.

Current space traffic coordination mainly relies on the U.S. Air Force's Combined Space Operations Center¹ (CSpOC) to provide state tracking and conjunction prediction emails to operational spacecraft. COLA strategies may be vetted by injecting proposed maneuvers into the ephemeris products supplied to CSpOC to assess conjunction mitigation. Space agencies of some countries have customized traffic management teams for their own satellites, e.g. NASA's Conjunction Assessment Risk Analysis (CARA) for NASA's non-human spaceflight missions and the French Space Agency CNES's CAESAR. Such Space Traffic Management (STM) services are centralized and cater to very specialized consumers, therefore *not* scalable to 10 or 100x increases in population. Moreover, any *new* STM system will have to account for existing players (CSpOC, CARA, etc.) and players that have supported them. For example, Space Situational Awareness (SSA) data is available from the U.S. Strategic Command and the Space Data Center (SDC) to their members. SDC is operated by Analytical Graphics, Inc. (AGI), a technology partner to the non-profit Space Data Association (SDA), which is a consortium of thirty major spacecraft operators, for-profit companies like LeoLabs, or databases of individual companies like Planet Labs that volunteer their information publicly. The SDC has scaled SSA in the last decade, and has APIs which allow new entrants to supply their data in their own native format (i.e. a Data Lake model). STM or SSA data is exchanged using message standards set by the Consultative Committee for Space Data Systems (CCSDS). Even within the U.S., regulatory authority to supervise and license commercial space activities is split across several agencies, e.g. NOAA for remote sensing satellites, FAA's Office of Commercial Space Transportation for launch and re-entry of spacecraft and their interaction with the national airspace, FCC for regulating communication spectrum allocation among commercial spacecraft applications. Operators are bound by domestic laws and regulations set forth by the Federal Communications Commission (FCC), National Telecommunications and Information Administration, Department of State, etc. In turn, such national regulators have to account for international law (e.g. the Outer Space Treaty), non-compliant players, and

orbital debris mitigation compliance with international forums like Interagency Space Debris Coordination Committee (IADC). Compliance is especially difficult to enforce outside of radio regulations by the International Telecommunication Union because no one body has regulatory authority over space (e.g. UN COPOUS is a deliberative body).

While groups like SDC have successfully integrated SSA data from multiple providers, and previous studies have discussed perspectives on the complex landscape of managing global space traffic [7], identified preventive strategies against dangerous overcrowding of low Earth orbit (e.g. slot based allocation of sun synchronous orbits [8]), and active mitigation strategies for preventing collisions against debris (laser-based orbital control [9,10]), no scalable automated solution for end-to-end STM or Space Traffic Coordination and Management (STCM) has been prototyped yet. As the space population scales, and conjunction alerts against growing number of debris have to be sent to a growing number of active spacecraft, the current system is inadequately equipped to handle the increased workload [11] and will benefit from better information technology. The recently signed Space Policy Directive 3 in the U.S. supports the transition of civil STM responsibilities to a civilian entity and provides additional guidance relevant to the development of an STM system. The International Association for the Advancement of Space Safety STM working group [12] and the AIAA STM working groups [13] have begun to identify the key requirements for such a system, as advised by academic, industry, legal and government stakeholders.

To keep the utilization of outer space safe, sustainable and efficient, to provide structure only when necessary and allow flexibility of operations otherwise, NASA Ames Research Center (ARC) has proposed an STM architecture - visually represented in Fig. 1, and introduced in detail in Refs. [14,15]. STM in this paper refers to the same concept as STCM. The architecture is a set of defined roles, standardized open interfaces (e.g. application programming interfaces/APIs), and data models (with required and optional fields, based on CCSDS and other industry standards), that allow automated and scalable interaction. The *defined roles* are:

- Owner/Operators (O/O), who own and fly satellites participating in the STM architecture.
- STM Service Suppliers (S3), who act as a concierge providing STM & compliance services to O/Os, serve as a link to the broader STM ecosystem, and procure services on behalf of O/Os.
- Space Situational Awareness (SSA) Suppliers, who are responsible for acquiring and fusing sensor observations with cooperative tracking data from O/Os to generate and maintain a catalog of space objects. SSA is a key example of an STM service that currently exists independently, therefore called out separately from S3.

¹ CSpOC was called Joint Space Operations Center (JSPOC) until July 18, 2018.

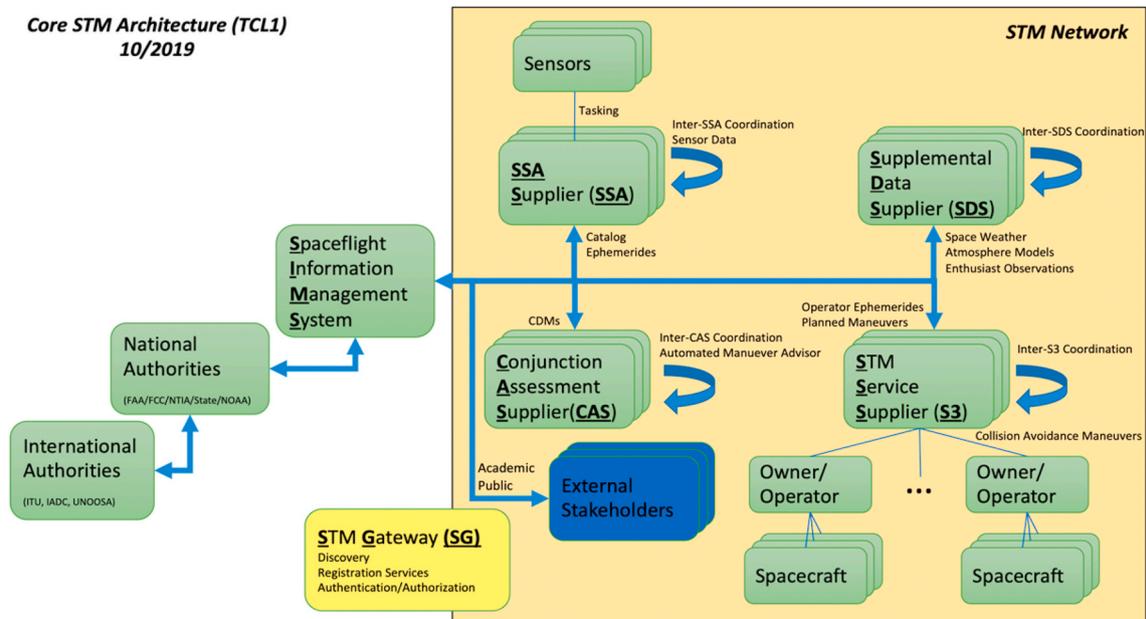


Fig. 1. Space Traffic Management architecture proposed by NASA Ames Research Center, as detailed in Ref. [14,15]. The Concept of Operations (ConOps) is based on NASA's successful UAS Traffic Management architecture that has been developed in partnership with the FAA.

- Conjunction Assessment Suppliers (CAS), who are responsible for screening objects against SSA catalogs for potential conjunctions, as well as verifying collision avoidance maneuvers proposed by S3s.
- Supplemental Data Suppliers (SDS), who provide other relevant data and services. Examples might include precision atmospheric modelling data to reduce errors in spacecraft orbit propagation or space weather warnings.

The roles and responsibilities are described in functional terms because the STM architecture makes no assumptions about the type of actor who would provide a given service (government, non-profit, or commercial) or regarding the separation of roles across actors. Multiple roles might be fulfilled by a single conceptual or legal entity. For example, a large O/O who flies many satellites might choose to act as its own S3 or contract with an outside provider. As space traffic gets more complex, new functions may be developed within an existing CAS or S3, or they may contract with an outside provider. For example, if a CAS has informed an S3 of an imminent conjunction and the S3 needs guidance on maneuver options and trade-offs, the S3 may use their in-house planning tools on CAS-provided data, request the CAS perform the analysis, or request an external service (e.g. Automated Maneuver Advisor (AMA) which automates tradespace analysis and maneuver advice) to use their custom planning tools on S3-provided CAS data related to the conjunction. New services added to the STM network may even be hardware services provided by entities that do not fall into any of the above categories (neither operators, brokers, software nor data suppliers). For example, de-orbit services may be supplied to operators of non-maneuverable spacecraft by external providers of such mechanisms (e.g. rendezvous chasers, tethers, robotics arms, lasers, aerodynamic decelerators).

The STM Gateway is a *management role* within the proposed architecture that will handle certain basic functions like registration, discovery, authentication of participants, and auditable tracking of data provenance and integrity. This open-access architecture allows any user to join the system, after reliable authentication, and be discoverable as a new participant by a centralized registry of participating entities or various decentralized discovery techniques. We can thus accommodate

users ranging from small academic Cubesats to proposed mega-constellations with thousands of satellites, while addressing their customized needs for communications, interoperability, regulation, and protection of proprietary data. The APIs and data models present a low barrier to entry for new or existing STM actors and/or services. As long as these interfaces are followed, STM is expected to serve as a non-hierarchical marketplace for services that can accommodate future regulatory requirements, and integrate data from multiple sources, providing information to those who need it to behave responsibly, while being responsive to source-imposed restrictions on sharing. Decentralization also implies that new nodes can be added *scalably*, and common standards allow software developed for one supplier to be *reused* and *interoperate* with another.

We have deployed a research platform within a simulation lab at NASA Ames. The goal is to visualize, assess, and validate our proposed STM architecture and performance under increasingly complex use cases, grouped into technical capability levels (TCL). The TCLs provide an evolutionary path to implement a software prototype of the proposed STM architecture and its application to groups of use cases. Our STM prototype is based on modern micro-service architecture adhering to OpenAPI standards and deployed in industry-standard Docker containers,² facilitating easy communication between different participants or services. Automation and autonomy will be indispensable in the efficient implementation of the architecture and its services. Automation in STM includes syncing multiple catalogs of resident space objects [14], while autonomy examples include rapid communication or negotiation between operators and cost-benefit trade-offs for spacecraft maneuver selection to prevent impending conjunctions between multiple active spacecraft. This paper describes the prototype and its application to on-orbit operations for a collision avoidance and maneuver planning use case, per civil/commercial catalogs. Once development is complete, the intention is to open source the platform code to enable use and further development by all stakeholders interested in participating in an STM

² Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

ecosystem, as well as eventual transition to the appropriate regulatory agency for use in operations.

2. Software prototype of the STM architecture

The STM software prototype implements the ConOps of our proposed architecture (Fig. 1) in successive TCLs, similar to the manner in which NASA's UAV Traffic Management (UTM) software prototype [16] was developed. UTM refers to the system that can integrate unmanned aerial vehicles (UAVs) safely and efficiently into air traffic that is already flying in low-altitude airspace, so that they do not interfere with helicopters, airplanes, nearby airports or even safety drones being flown by first responders helping to save lives. UTM is currently being handed to the FAA for operational use via the NASA UTM Research Transition Team that was formed to 'collaboratively explore concepts, develop prototypes, and demonstrate a possible future UTM system to enable large-scale low altitude UAS operations' [16]. Borrowing insights from UTM's success, the proposed STM ConOps is aimed to integrate existing vehicles and new orbital entrants safely and efficiently into the space above the Von Karman Line.

We used the iterative Software Development Life Cycle (SDLC) process to develop the STM prototype. SDLC's phases included creating a detailed plan for our requirements analysis, definition, product design, architecture, development, testing and deployment. This section presents our implementation of the system software based on the proposed STM architecture (each role is an 'application') and describes the functional elements and their interactions using activity diagrams and API structures. A prototype implementation is demonstrated from the front end perspective (for all roles in Section 1) in the following video: https://sreejanag.github.io/Videos/AmesSTM_demo_2019.mp4

The STM architecture requires applications (S3, STM Gateway, CAS, SSA) to interact as loosely coupled services, making the microservices architectural pattern a strong fit for STM implementation. Microservices allow the construction of small independently versioned and scalable customer focused services with specific business goals which communicate with each other over standard protocols with well-defined interfaces. As the services are independently deployable and scalable, each service also provides a firm module boundary. Other benefits of microservices have been listed in Appendix A.

2.1. STM software stack

The STM software stack is an implementation of our proposed STM architecture, intended to provide the basic *functions* (yellow box in Fig. 1) and serve as a portal to allow seamless interaction of *roles* and *services* (green boxes in Fig. 1). It was designed to be simple, highly extensible and modular, while utilizing as much open source technology as possible. The main components of the stack are listed below, and the selected technology used for each component italicized. The public software developer ecosystem was studied to choose stack technologies (comparison and selection described in Appendix B) which would be easy to implement, and is well supported by a reputable open source community.

User Interface (UI) webpages have been developed for the STM gateway (SG) and S3. The SG UI currently supports basic functions of registration and discovery, providing capabilities for STM managers to manage and verify interfaces between S3-CAS-SSA. The current S3 UI supports O/O registration, setting up screening requests to the CAS, and displaying results from the CAS. We selected *Angular 5*, a modular and extensible web and UI development framework to provide capabilities such as notifications, integration with web exchange protocols. For the web server technology, we chose *Express* due to its extensive routing features including routing, separating handlers (put, get, post, etc), static file serving, and a framework that many popular template engines can plug into.

Application Program Interfaces (APIs) are an effective interface

mechanism to share functionality amongst independent applications/programs while maintaining code separation. STM APIs have been developed to provide interfaces to CAS, SSA, S3 and SG, using simple http protocols. Our requirements for API were to develop a door/inlet into each service (CAS/SSA/S3/SG), allowing other programs to securely interact with it without the need to share any code. *Node JS*, a Javascript-based runtime environment that remains lightweight and efficient in the face of data-intensive real time applications, was chosen as the preferred API framework. We use SwaggerHub, an API development platform that leverages the core capabilities of the open source Swagger framework to build, document, manage, and deploy the STM APIs.

Databases (DB) were created for storing user authentication and authorization, storing requests, responses and configuration details. Our requirements from a DB were a simple setup, to be Docker friendly (to deploy easily, as described in Section 2.2), allow JSON-like binary data points (BSON). *MongoDB* was chosen as the database technology because it met the said requirements in a scalable and flexible way. It is open source, document-based, and can change streams to push updates across and from various service providers and STM roles.

To summarize the prototype implementation of *functions* and *roles* within our proposed STM architecture: SG and S3 have been developed in Angular (UI) and Express (web server). Their sequence diagrams and functions will be explained further via the use case in Section 3. The example CAS has been developed in MATLAB and Analytical Graphics, Inc's Systems Tool Kit (STK) software, and its structure is described in Ref. [17]. Its utilization for complex maneuver planning will also be explained further via the use case in Section 3. The example SSA draws primarily from the space-track.org database that lists the most recent Two-Line Element (TLE) ephemerides, as provided by CSpOC as a free service to the public and satellite operators. We are currently testing UT Austin's integrated database³ [12] of various SSA sources (including CSpOC) as an alternative SSA example. The S3, CAS, and SSA are hosted on separate servers to mimic disjoint entities within the STM network. Currently, we have implemented an example AMA in Python inside the CAS application, on the same server. However, since it exchanges information with CAS using pre-defined APIs, the AMA and CAS can be separate entities as well. Fig. 2 shows the STM test bed in terms of functions within the prototype implementation, while Fig. 3 shows the chosen technologies. Deployment and information flow sequence will be described in Section 2.2 and 2.3, respectively.

2.2. STM stack deployment

The STM stack was developed and deployed into Docker containers in the microservices architecture. Docker containers are lightweight compared to virtual machines making them resource effective. They provide uniformity across development and production environments, making deployment agnostic to the operating system. They are suitable for continuous integration and deployment, with consistency across multiple development and release cycles, and have an excellent support base with several official supported containers. The Dockerized, and thereby standardized, STM environment has allowed repeatable development, build, test and production.

The current STM stack as deployed is illustrated in Fig. 3. While it can support multiple S3/CAS/SSA or other services, per the STM architecture and marketplace described in Section 1, only one instance of each has been shown. The stack was verified on Windows 10, Mac OS 10+, and Ubuntu. NASA's stringent security requirements prevent admin access on any host machine. To work within NASA security parameters, which is perhaps representative of siloed application restrictions that STM is expected to operate in, the containers are deployed

³ AstriaGraph page accessed on October 2, 2019: <http://astria.tacc.utexas.edu/AstriaGraph>.

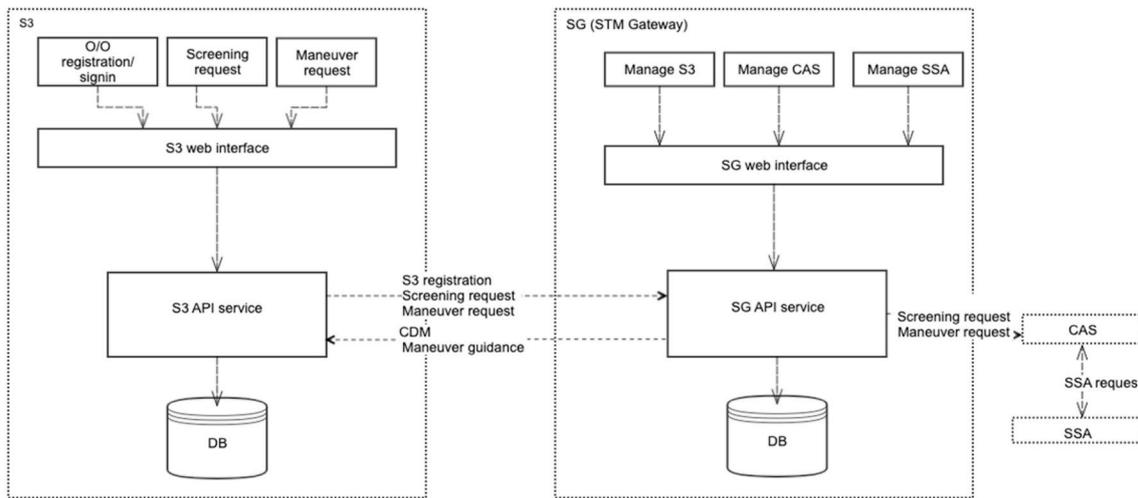


Fig. 2. Current STM Testbed showing the functions for SG and one S3 instance, interacting with one CAS and one SSA. The architecture supports multiple instances of services (S3, CAS, SSA, etc.).

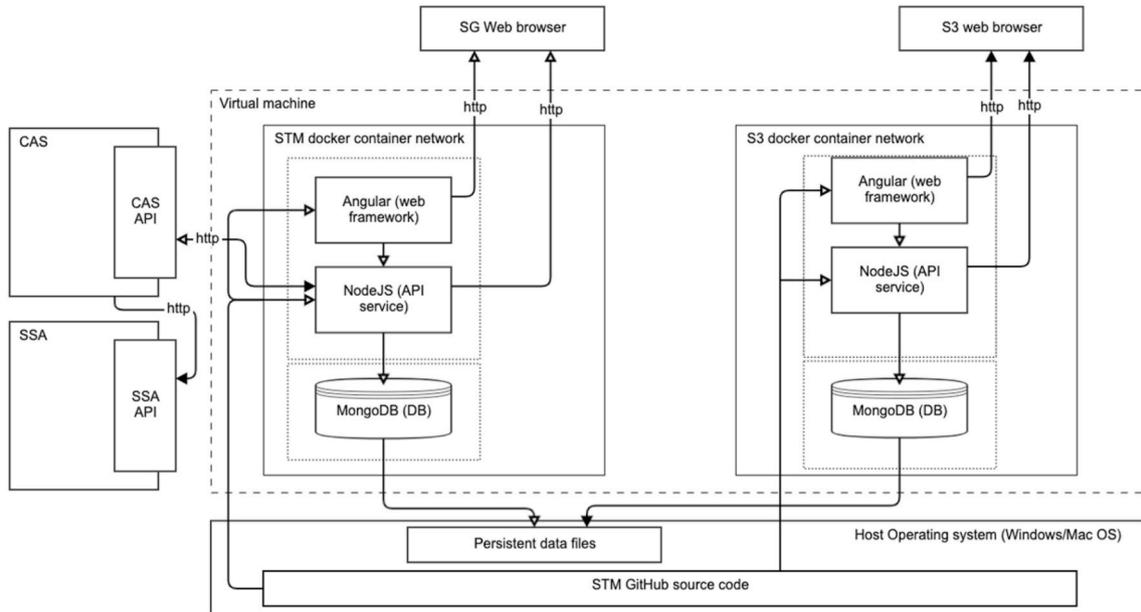


Fig. 3. Current deployment structure of the STM stack. Arrows represent flow of information. Only one instance of services (S3, CAS, SSA) is shown, however multiple are possible.

inside a Virtual Machine (VM). We used Ubuntu 14 as the VM to deploy the application containers. SG and S3 applications were available over http in a web browser, and will be extended to https for future implementations. SG and S3 have two containers each: MongoDB, Express + Node. The hosted Express servers which rendered the Node. JS API and the compiled Angular app were rendered as HTML + JS pages. Data persistence is achieved by configuring the MongoDB to allow the SG and S3 to store the data files on the host machines.

2.3. APIs and interactions

Since STM’s core benefit is the availability of common APIs and data models that roles can use to interact with each other seamlessly, we defined them on SwaggerHub. Every piece of information exchanged

between two containers or servers in Fig. 3 would invoke a Swagger request that adhered to defined APIs. For example, the OpenAPI model of the SSA response to an orbit data request is seen in Fig. 4. In this example, the SsaOrbitDataMessage model consists of the SsaRSO (“Resident Space Object”) field, which specifies the catalog used and identifier within the catalog, and the SsaODM (“Orbit Data Message”) field, which contains the ephemeris data for the object. This information is transmitted as a JSON object. The nature of the pull requests to the SSA catalogs and publication/subscription of data will be sensitive to the regulations of the data sources. For example, the ingestion of SDC data can be regulated by the SDA and available only to members, data from commercial providers like ExoAnalytic or LeoLabs can be subject to a fee, and advanced service access CSpOC data may need an SSA sharing agreement with the United States Strategic Command [15]. Once a

```

SsaOrbitDataMessage{
  SsaRSO* RsoID{
    description: Resident space object identifier and catalog

    catalog string
              example: NORAD
              Enum:
              Array [ 3 ]
              string
              example: 27386
  }
  SsaODM* RsoOrbitData{
    description: CCSDS-like Orbit Data Message (state vector based)

    epoch* TimestampUTC string($date-time)
            ISO8601 (UTC) timestamp string.
    frame* string
            example: EME2000
    position* [...]
    velocity* [...]
    covariancePV [...]
  }
}

```

Fig. 4. Example OpenAPI model of the SSA response to an orbit data request.

connection is authenticated, APIs of varying complexities (very simple example in Fig. 4) can be used to exchange data in keeping with in keeping with National Space Traffic Management Policy’s Open Architecture Data Repository, and other existing standards by CCSDS. Reference [14] Section V-C describes proposals for automating catalog access. Similarly, all interactions between service providers and with SG can occur through such APIs and have been implemented in our STM prototype as described below.

The SG manages the basic STM functions (e.g. registration, discovery) that allow interactions between S3, CAS and SSA. Each S3 can register with the SG using a self-generated UUID identifying it uniquely. It stores CAS, SSA and S3 configuration settings via the S3 + SG API in MongoDB (Fig. 3). STM’s core customers are the spacecraft O/O’s, and any S3 supports O/O registration and O/O sign-in. The S3 login page is shown in Appendix C (#1).

Fig. 4 is a sequence diagram of typical interactions between the STM service providers for screening and collision avoidance services. S3 may invoke a screening request for the O/O’s spacecraft at a pre-determined frequency or on an as-necessary basis, as seen in the UI screenshot in Appendix C (#2). The request specifies the O/O’s unique ID and properties. The UI also allows the S3 to enlist their choice of SSA and CAS provider for the request. Appendix C (#2) also shows the S3 discovery of CAS and SSA registries via APIs. These registries are published by the SG.

Once the screening request is submitted by S3, it invokes the S3+SG API for user authentication, authorization, storing and retrieving screening requests and responses, and then invokes the CAS API. These APIs are exposed securely by the SG. The CAS server, upon receiving the request, invokes the SSA server using the SSA API to retrieve the most updated TLE and state information on all spacecraft. The SSA server log can be seen in Appendix C (#3). Note that this workflow assumes that the CAS, SSA, CAS-AMA servers are set up (running and ready to receive) before the S3 launches its requests.

Upon receiving the SSA response via API, the CAS server initiates the screening process using an example conjunction assessment application running on proprietary software (in our case, MATLAB and AGI STK). Appendix C (#4) shows the screenshot of this step. The CAS app processes the screening results received from MATLAB and STK, then invokes an API to return Conjunction Data Messages (CDM) to the S3. Each potential conjunction is selectable on the UI, so that the S3 can identify

High Interest Events (HIE) among the CDMs that warrant more analysis and re-submit to CAS to re-run with more accurate SSA sources and to observe how the event evolves over time. Accurate observations are fundamental to planning for space safety [18], therefore the ability to task additional observations based on covariances for a potential conjunction is an important assurance feature in STM. The current prototype allows the S3 to select HIEs from the list of CDMs that they would like to plan maneuvers for. Appendix C (#5) shows a screenshot of the API as visible to the user in this step. Submitting the planning request (‘Get maneuvers’) invokes the CAS API, which then calls the AMA tool (currently housed inside CAS).

Once the maneuver request is submitted by S3, it invokes the CAS-AMA API which starts the AMA application and runs the maneuver planner. An example scenario will be described in Section 3 showing the iterative nature of interactions (specifically in Section 3.3) between AMA, CAS and SSA in scheduling maneuvers for multiple satellites involved in frequent conjunctions. After completion, the AMA responses are currently stored on the CAS server and may returned to the S3 in the form of graphs and data files characterizing the ‘proposed maneuvers’. If more than one maneuver sequence is proposed, the S3 may recommend one ‘accepted maneuver’ in consultation with the O/Os, following which the O/O makes the orbit correction, and the S3 pushes the updated orbit ephemeris to the SSA. In turn, the SSA makes the information available to the wider STM network so that every other operator now has access to updated information. Throughout the process, security between S3 and SG is maintained with a combination of token and UUID for each S3. MongoDB stores the user(s), screening request and response, system logs, and interacts with the other applications using the S3 + SG API to MongoDB (Fig. 3).

3. Application to sequential conjunctions and planning collision avoidance

The STM prototype was used to test conjunction assessment and COLA use cases that represent complex, cascading behaviour. Current state of the art is that spacecraft operators get an email from CSpOC or another automated SSA/CAS service informing them of future conjunctions involving their spacecraft, and if they have maneuvering capabilities, they execute a COLA maneuver. There is rarely coordinated

planning (or even communication exchange) between operators, and at the current orbital population, rarely a series of consecutive conjunctions expected for the same satellite in a few days. Therefore, realistically based on conjunction uncertainty and operator preferences, doing nothing may often be the least risky (and least expensive) path. However, rare occurrences of mischaracterization have led to catastrophic and near-permanent damage in space. For example, the Iridium 33 collision with the defunct Kosmos 2251 in 2009, resulting in >2000 pieces of debris >10 cm, was caused due to uncertainty in SSA data, inadequate methods of orbital determination, and the absence of wide data sharing that limits planning under uncertainty [19]. More recently, ESA's Aeolus satellite maneuvered to mitigate a conjunction with a SpaceX satellite in September 2019, after failed communication with SpaceX when PoC grew rapidly in the days before the conjunction, and ensuring a final miss distance of 306 m [20]. As SpaceX increases its Starlink constellation size to 7500+ (albeit maneuverable) satellites [21], such conjunctions are expected to become more frequent, and COLA maneuvers may have imminent, sequential impact on following conjunctions in a 'cascading' fashion. While the UN Committee on the Peaceful Uses of Outer Space guidelines for enhanced registration makes it easier for operators to get in touch, an automated pipeline for S3 communication, negotiation, and decision-making will improve efficiency across constellations owned by disjoint entities.

Our proposed STM architecture allows participants to pull from continuously updated individual SSA sources or well-integrated groups of sources (example format suggested in Section 2.3), and to publish O/O provided information automatically to such sources (start and end of sequence in Fig. 4). We have developed an example CAS [17], that is capable of automated (A) conjunction screening, i.e. 1-vs-all or N-vs-M conjunction predictions, (B) encounter identification to flag HIEs from the conjunction list based on pre-determined threshold probabilities of collisions (PoC), covariance errors allowed, etc., (C) maneuver generation to list possible COLA strategies to avoid the HIE, and (D) maneuver screening, i.e. conjunction assessment for the resultant satellite orbit if the O/O were to execute any of the COLA strategies. CAS interacts with S3 via standard APIs to provide CDMs; S3 chooses HIEs among the CDMs and submits a maneuver planning request to CAS (Fig. 4). Reference [17] describes the example CAS, the algorithms used to enable the above A-D steps, and the trade-offs between the COLA maneuvers generated for consideration. This paper extends the CAS capability to introduce planning and scheduling a sequence of efficient maneuvers to clear conjunctions by maximizing a reward function that factors in system-wide conjunction risk among an arbitrary set of satellites. The maneuver planner is currently internal to the example CAS ('generate proposed maneuver' in Fig. 4), however it can be a separate service that uses data exchanged between the CAS and SSA. The CAS returns a list of proposed maneuvers to S3, which selects a maneuver and implements it in consultation with the O/O, after which the new TLEs are updated in STM (Fig. 4).

To demonstrate the capability of the STM prototype in a COLA sequence, we conducted a case study involving sequential conjunctions between six active satellites over a one-week period, overseen by an S3. Current state of the art maneuver planning executes steps A-D in sequence and is sufficient for infrequent COLA decision-making by a single active satellite against debris or non-cooperative satellites. It may also be extendable to multiple conjunctions between that satellite and the already identified piece(s) of debris. Maneuver planning for multiple controllable agents with frequent conjunctions is more complex; it requires steps A-D to be executed iteratively as an algorithm searches through the planning horizon for an optimum sequence of maneuvers. This case study demonstrates the STM prototype's ability to seamlessly

exchange information among various service applications on different platforms and servers, in keeping with process flow in Section 2.3, to enable *continuous* and *iterative* collision prediction and control. While the example planner is a system-wide reward-maximizing, greedy scheduler that outputs a schedule for COLA actions by each satellite, it paves the path for new research in satellite maneuver planning, scheduling and negotiation across disjoint entities, enabled by the availability of an STM network. It is extendable to conjunction scenarios with non-maneuverable satellites or non-cooperating actors or debris (which are assumed to have right of way) by selecting the active maneuverable satellites and optimizing for rapid communication, negotiation and maneuver planning among them, given known constraints.

3.1. Conjunction screening and encounter identification

For a one week period starting May 20, 2010, we simulate an S3 that is responsible for 6 spacecraft (NORAD ID # 10676, 25419, 25477, 41918, 42809, 42961) belonging to one or many operators, henceforth called *primary* spacecraft. When sending the screening request to the CAS, the S3 sets the "do-not-violate" thresholds on PoC to be $1e-6$ and the spacecraft error ellipse to be $20 \text{ km} \times 10 \text{ km} \times 5 \text{ km}$, in the along track, cross track and radial directions. While this PoC is more conservative and the error ellipse far larger than typical, we set it so that several sequential conjunctions are detected and collision avoidance maneuvers have a good chance of inducing new additional high-risk conjunctions over the study period, without having to forcefully increase the satellite population beyond the current database. The aim is to mimic a scenario when the population is much larger, and therefore where sequential conjunctions are commonplace, and therefore there is significant risk that moving will make conjunctions worse and cost fuel to the operator satellite.

While true PoC is a more accurate metric to assess risks, it is computed by integrating a three-dimensional probability density function that quantifies the uncertainty of the relative position between any two satellites. Since we did not have access to the covariances of the 6 primary satellites, an alternative measure of risk called maximum PoC (Max PoC) was used, which is the worst-case PoC possible for a given conjunction geometry. The CAS performed the request by executing MATLAB code that automated an instance of AGI STK and AdvCAT, an add-on conjunction analysis tool. A one-versus-all screening is carried out, i.e. the orbit of each of the 6 primaries is compared to the orbits of all remaining RSOs over a week of simulation. For simplicity, STK's HPOP propagator was used with TLE-derived state vectors as initial conditions.

Based on these parameters, ~50 conjunctions or encounters were returned as CDMs by CAS to S3, and are referred to as *primary conjunctions*. Encounters identified are those whose Max PoCs violate the PoC threshold during the scenario time period.

3.2. Maneuver generation and screening

Candidate COLA maneuvers are generated to mitigate the primary conjunctions using the analytical formulation of Bombardelli and Hernando-Ayuso [22], implemented on MATLAB. Given an impulse budget ΔV and maneuver location $\Delta \theta$ before the Time to Closest Approach (TCA), this algorithm finds the optimal impulsive ΔV vector orientation that maximizes the miss distance between two objects at TCA. Future work may optimize the in-track, radial, cross-track directions of thrusting such that COLA actions may be incorporated into regular station-keeping (e.g., prograde in-track burns are almost free for a circular orbit). As with other elements of the CAS, note that it can

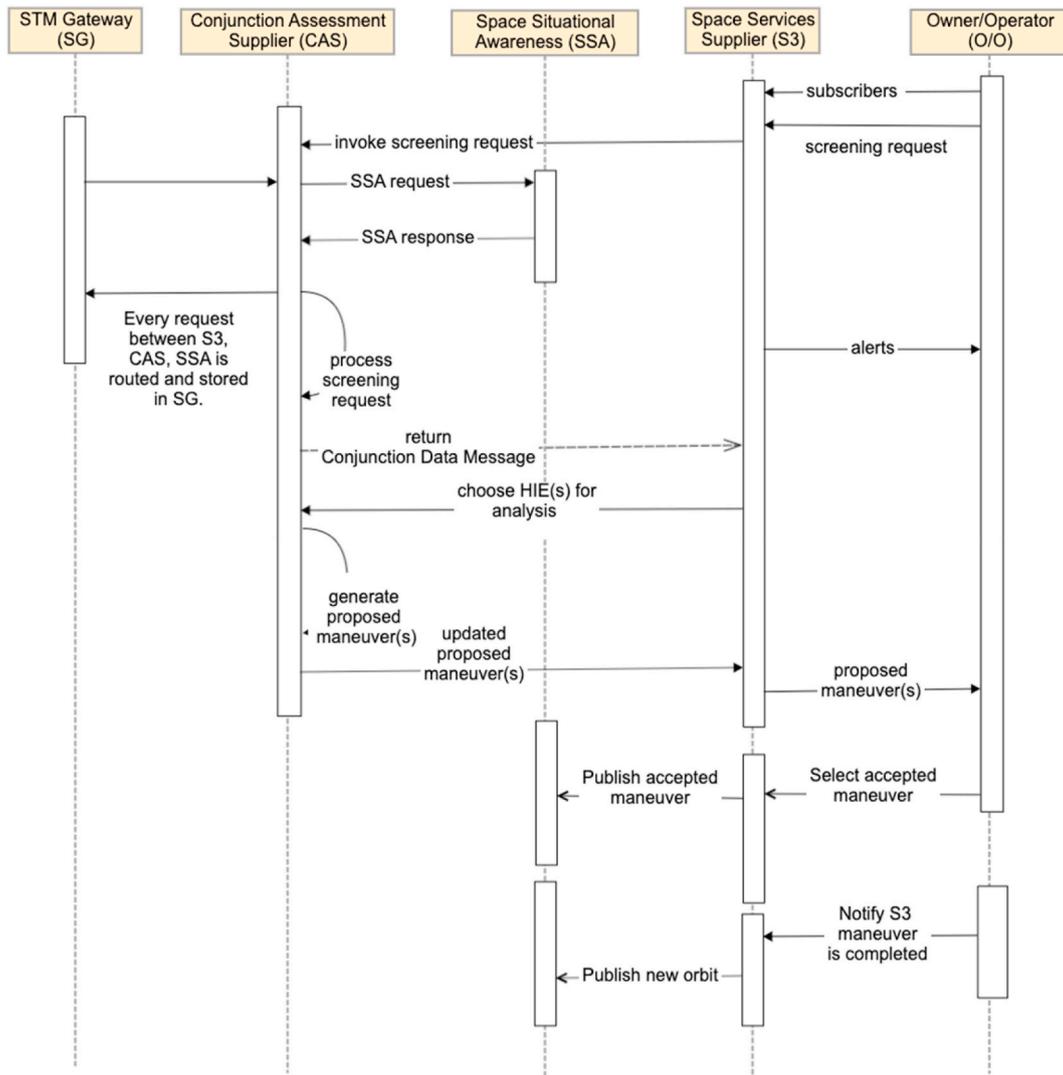


Fig. 5. Sequence Diagram for interactions between the STM Gateway and various roles/providers in the STM network for the screening and conjunction assessment workflow.

replaced by another algorithm, should it be more appropriate.

The goodness of a maneuver is determined (screened) not only by the extent to which the primary conjunction was mitigated, but also by additional conjunctions that the maneuver causes. A *secondary conjunction* is a post-maneuver conjunction between the primary satellite and the same secondary RSO that resulted in the HIE, which motivated the maneuver. A *tertiary conjunction* is a post-maneuver conjunction between the primary satellite and a completely different RSO.

The maneuver planner is informed by insights from several case studies, three of which are documented in Ref. [17] - a HIE between an active satellite and debris object, an active satellite against four others, a head-on conjunction between two satellites in very similar orbits. The results show that primary conjunction mitigation and minimizing new secondary/tertiary conjunctions are conflicting objectives even for a single maneuverable satellite to avoid a single HIE (Figs. 5, 7 and 8 in Ref. [17]). Therefore, an inclusive objective function is required for optimum planning. For multiple sequential conjunctions, at any given point in time before TCA, there may be multiple maneuver choices by a satellite, depending on which conjunction it is trying to avoid. Fig. 5 shows candidate maneuvers by a satellite (NORAD #15333) in order to

avoid imminent collisions with four RSOs (in four colors, where the legend contains their NORAD IDs). The original Max PoC of each event is shown as dashed lines; the TCA for each event corresponds to the point where the dashed and solid lines intersect. The markers represent the reduced Max PoC if a $\Delta V = 1$ m/s maneuver is executed at the said epoch. Between 22 and 26 h after the screening epoch, NORAD #15333 may choose to execute the red or green maneuvers in order to avoid NORAD #41343 and #36155 respectively. Therefore, each time step in the planning horizon can be considered a Markov Decision Process (MDP) for the satellite #15333. In the figure, screening epoch is defined as the simulation start time.

While this example shows the importance of multi-objective optimization (reward proposed in Equation (1) may be used) to select a maneuver or a combination, the controllable, decision-making agent is still a single S3 managing a single spacecraft. For generality, the main case study in this paper shows sequential conjunctions between multiple *controllable* spacecraft and involves a maneuver tradespace for every spacecraft, and every conjunction. The O/O(s), mediated by their S3(s) and informed of the tradespace by their CAS, are expected to consult, plan and decide which maneuver is to be selected.

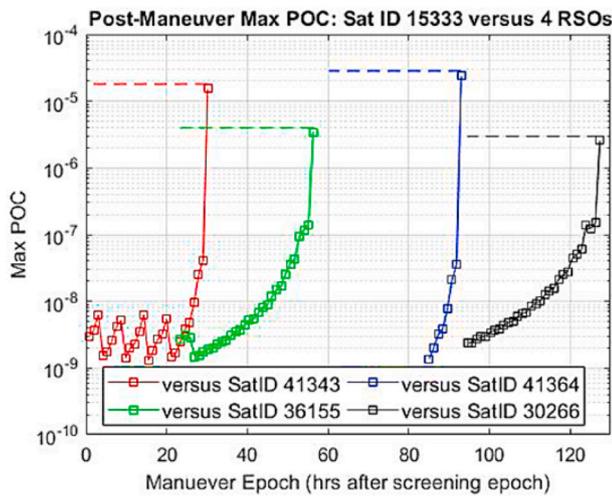


Fig. 6. Example of sequential conjunctions and the maneuver options (varying $\Delta\theta$ only) generated for COSMOS 1603/NORAD ID: 15333 to mitigate them [17]. Maneuver performance is quantified by the post-maneuver screening, in terms of Max PoCs against the four spacecraft that caused the primary conjunctions. Each square is a 1 m/s maneuver (ΔV) arranged by execution epoch ($\Delta\theta$).

3.3. Maneuver planning

An example system-wide reward function for maneuver planning is proposed in Equation (1). In the presented case study, Equation (1) is evaluated for a set of potential collision avoidance maneuvers on a per-HIE basis, but this function could also be cumulatively added over time-steps to support alternative planning algorithm designs. It maximizes mitigation benefit minus cost across all controllable spacecraft, so that the space ecosystem is safer and more efficient as a whole. The process of implementing it across disjoint entities is the credit of the STM architecture and its software prototype, per the process in Section 2.3.

$$\begin{aligned}
 ptr = & w_1 * s_1 * [new_prim_PoC * \{(new_prim_PoC - th * 10^{-2}) > 0\} - old_prim_PoC] \\
 & - w_2 * s_2 * \Delta V \\
 & - w_3 * s_3 * \sum sec_PoC * \{(sec_PoC - th * 10^{-2}) > 0\} \\
 & - w_4 * s_4 * \sum tert_PoC * \{(tert_PoC - th * 10^{-2}) > 0\} * \frac{1}{\log(timeToConjunct)} \\
 & - w_5 - P
 \end{aligned} \tag{1}$$

$$P = -\infty \text{ if } PoC(t + \Delta t) > th0, \text{ otherwise} \tag{2}$$

The first term in Equation (1) captures the extent of mitigation of the primary conjunction, ignoring conjunctions with max PoC two orders of magnitude or more below the HIE threshold PoC. The second term captures the required delta-V of the maneuver in m/s. The third term captures the total PoC of secondary conjunctions introduced due to the maneuver, again disregarding conjunctions with max PoC two orders of magnitude or more below the HIE threshold. If a maneuver changes the time of the primary conjunction but does not eliminate it, this will be recorded as a secondary conjunction. The fourth term captures the total PoC of tertiary conjunctions introduced due to the maneuver (those with an RSO not involved in the mitigated conjunction). These conjunctions are also filtered to remove any more than two orders of magnitude below the threshold max PoC and additionally weighted by the time until the tertiary conjunction, so that conjunctions with the entire space

ecosystem that are far into the future do not overshadow immediate ones. A log is used to reduce the severity of fall-off in the value of future conjunctions. The fifth term is a constant cost to plan any maneuver, so that a single maneuver is preferred to multiple ones to mitigate potential impact to the primary mission, all else being equal. This term is irrelevant for the implemented greedy per-conjunction algorithm, but is important for state-space search algorithms to capture practical preferences; e.g. operators prefer to advance or defer pre-planned station-keeping maneuvers if they do decide to move their spacecraft for COLA. The sixth term (P) is a policy term that is set to $-\infty$ if any PoC (new primary PoC, or any secondary or tertiary PoC) is greater than the threshold PoC that flags an HIE, within a pre-defined Δt , because any maneuver that creates a new highly probable conjunction within a short horizon is deemed unacceptable. For the greedy algorithm, no Δt was used, simply requiring all HIEs to be cleared before they occur. The weights w_x are set by the S3, in keeping with their customer/operator's priorities of the above described parameters. All were set to unity for this case study. The scaling factor s_x for every term is automatically determined by the planner, based on the available set of maneuvers and their corresponding PoC, so that each term is of similar order of magnitude. Δt was set to two orbital periods. The reward function can take positive or negative values, with higher reward being better.

Running a full factorial of maneuvers for all 6 satellites with the combinatoric of variations in the executing satellite, ΔV of the maneuver and the maneuver execution time (corresponding to $\Delta\theta$ in the orbit when ΔV is applied) is computationally unrealistic because it entails re-computing Equation (1) (i.e. re-propagating RSO orbits, and doing a computationally expensive search for conjunctions) thousands of times. Simulated annealing (a traditional global optimization method) with the above reward took 3–12 h to find the optimum maneuver sequence for sat #25419 alone, with $\Delta V = 1\text{--}10$ m/s, a week of temporal search space and various combinations of tuning parameters. Algorithm complexity scales with the power of the number of satellites. Therefore it was deemed unfeasible for exploring multiple controllable satellites, primarily owing to the computational load of the propagation and AdvCAT screening steps.

Instead, we devised a Multi-Spacecraft Maneuver Advisor Algorithm (MSMA) that informs the example AMA. The MDP is formulated as a graph-based search space for every spacecraft [23] where the decision variable at any time step is a maneuver option, and the optimum path traces the time series of these maneuvers within feasibility constraints. MSMA has been implemented in Python and is housed within the CAS. It is a greedy algorithm to clear all conjunctions involving at least one of S3's 6 spacecraft over the scenario time period. For each HIE in chronological order, the algorithm (1) screens for potential conjunctions above an S3 or operator-specified risk threshold per Section 3.1, (2) generates a tradespace of COLA maneuvers for all S3 spacecraft, per Section 3.2 by varying ΔV and $\Delta\theta$ between an HIE-clearing maneuver and next HIE; (3) calculates the reward for each candidate COLA maneuver per Equation (1) with PoCs computed per Section 3.2; (4) selects and simulates the COLA maneuver with the highest reward; (5) screens for HIEs and repeats this process until the system is free of HIEs from the start to end of the case study time period.

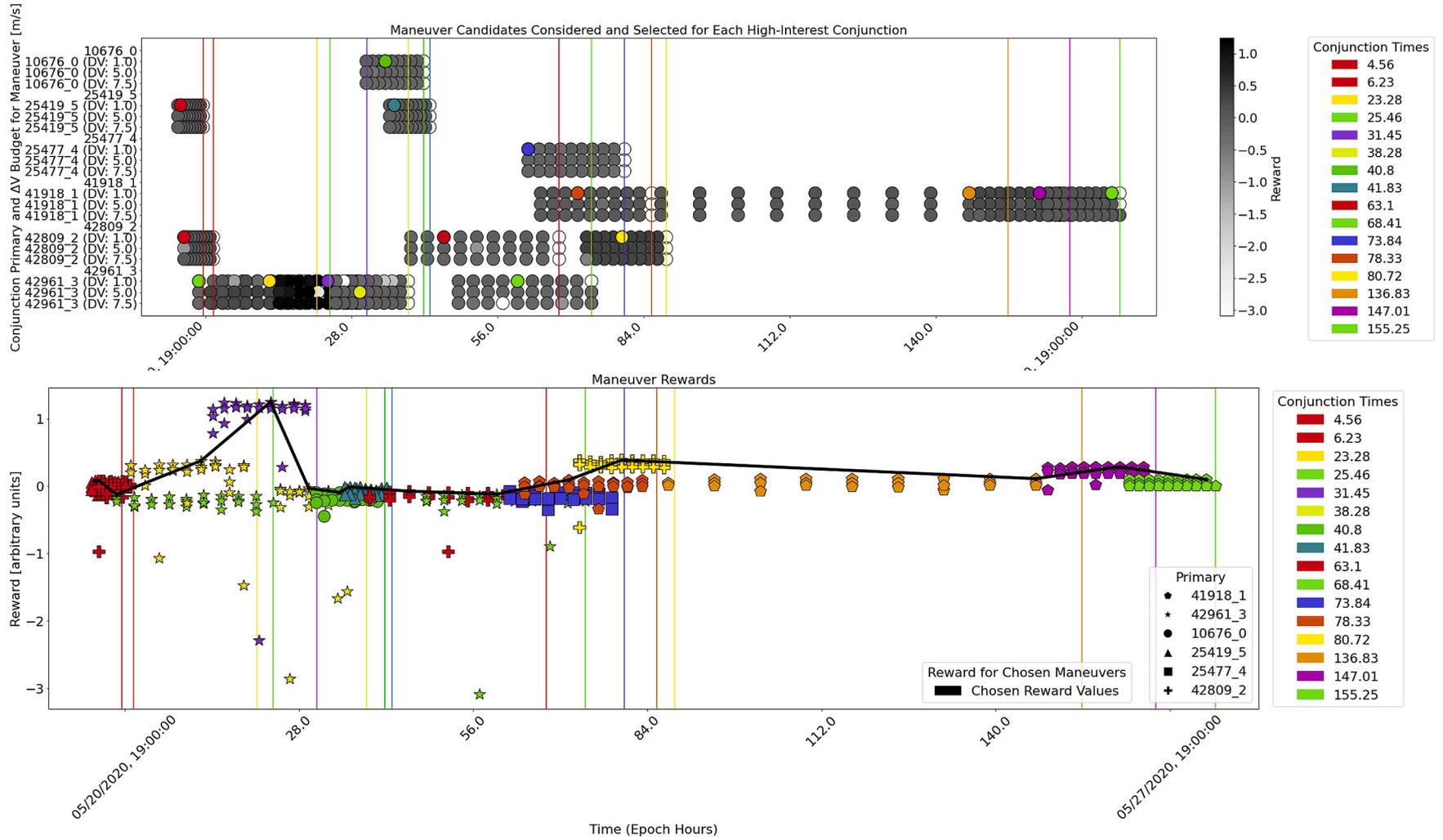


Fig. 7. [top] Full tradespace of maneuvers generated by the planner, as a function of time of execution time (X-axis) and executing 6 satellites per the listed ΔV in km/s (Y-axis). The 13 conjunctions mitigated are indicated by vertical colored lines at TCA (label in bottom). The candidate maneuvers are shaded in greyscale in proportion to their reward. The chosen maneuver to mitigate a conjunction is shaded in the same color as the conjunction vertical it mitigates. [middle] Maneuver tradespace with their reward on Y-axis. The chosen maneuvers are joined with a black line and the cumulative reward (by integrating Equation (1)), obtained by adding up the reward associated with each maneuver to mitigate each conjunction. The maneuver markers and conjunction lines are color-matched throughout. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

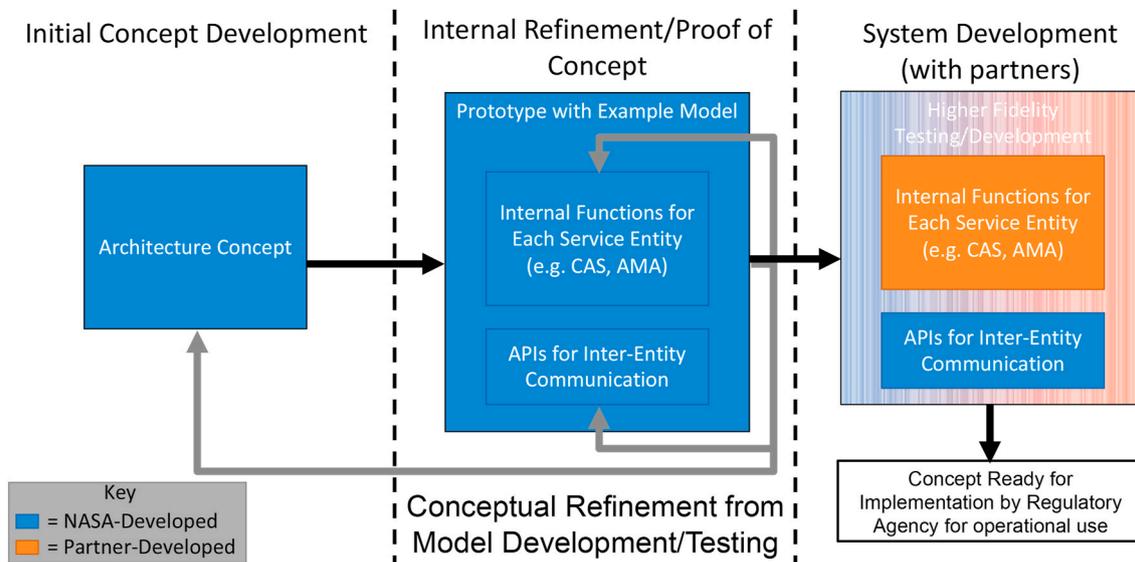


Fig. 8. Development cycle of the STM prototype from the proposed architecture on the left (past work), to a software prototype which uses sample models or examples developed in-house as service entities in the middle (current state), to a higher fidelity software prototype which uses partners to represent service entities (future work).

Running the MSMA for the case study duration of 1 week identified and cleared sixteen conjunctions that included at least one of the six S3’s satellites and warranted a COLA action. Some of these conjunctions were present in the original scenario and some new ones were introduced by the proposed COLA maneuvers. Nonetheless, MSMA in the example AMA implementation reduced the total number of HIEs from ~50 (screened in Section 3.1) to 16, and cleared all of them by distributing the COLA maneuvers across all 6 spacecraft. Fig. 6 shows the spread of the 16 mitigated HIEs as vertical lines of different colors, whose abscissa corresponds to their TCA epoch within the evaluated week. The units of the X-axis and conjunction time legend are in hours since 2020/05/20 7pm UTC. Each conjunction is assigned a unique color; those that seem visually similar have been placed spatially apart to ease readability. Fig. 6 also shows the maneuvers considered and recommended by the MSMA to mitigate the said 16 conjunctions. The full list of conjunctions, TCAs, and the satellites and maneuvers selected to mitigate them are listed in Appendix D.

For every conjunction from left to right in Fig. 6-top, MSMA generates M*N maneuvers per maneuverable satellite; across M propellant options (ΔV) and N epochs ($\Delta\theta$) evenly spanning the time-space between the maneuver selected to avoid the previous conjunction (colored circles in Fig. 6-top) and the next conjunction (Fig. 6-vertical lines). The figure shows the full tradespace of M*N maneuvers considered per conjunction as circles, where the parameters of each maneuver are listed in the Y-axis (satellite NORAD ID and ΔV) and arranged by execution epoch ($\Delta\theta$) in the X-axis. Colored circles represent the maneuvers that are recommended by the MSMA, assumed to be selected and executed to prevent the conjunction (vertical line) of the same color. For non-selected maneuvers, the darker the circle, the better that maneuver is according to Equation (1). Maneuvers that cannot clear a conjunction are assigned $-\infty$ reward and plotted as open circles. One maneuver option (ID# 10676, $\Delta V = 7.5$ m/s, last $\Delta\theta$) has been removed from the plot since its extremely low reward of -14.45 biases the greyscale. While $N = 10$, $M = 3$ in the results presented, they can be easily changed to improve

fidelity at the cost of increased runtimes.

While conjunctions may get added later (toward the right of the figure) due to maneuvers implemented to clear earlier conjunctions, it is possible that a particular maneuver mitigating a conjunction will induce a new conjunction *after* the particular maneuver but *before* the TCA of the mitigated conjunction. This happens in Fig. 6 when the third conjunction from the left (green) is mitigated resulting in another conjunction (yellow) with a TCA before the TCA for the green conjunction. Each of the resulting conjunctions are therefore also planned for and mitigated in sequence. The temporal flow of the maneuvers is from one colored circle to the next, starting from the left and moving rightwards.

The MSMA recommends the maneuver that clears the most imminent conjunction with the highest cumulative reward (maximizes Equation (1)), before planning for the next conjunction. Fig. 6-bottom shows the reward value of all potential maneuvers considered to avoid the 16 conjunctions. The shape of the markers corresponds to the primary satellite that will execute the potential maneuver. For every marker shape (i.e. satellite), the color of the marker represents the conjunction (same colored vertical line) that maneuver is intending to avoid. There are M*N markers per shape and color to account for the 30 maneuvers in the tradespace per conjunction (for $M = 3$, $N = 10$). A thick black line connects the markers representing the selected maneuvers to be executed. Initially, a maneuver involving satellite 25419_1 is selected to mitigate the conjunction at time 4.56, then satellite 42809_2 moves to clear another conjunction at time 6.23, followed by Satellite 42961_3 moving to clear a third conjunction (green) at time 25.46. This produces another conjunction (yellow) at time 23.28, which is again cleared by a maneuver. This continues until no conjunctions exist over the planning period. Candidate maneuvers that attain $-\infty$ reward are not plotted.

While none of the 16 conjunctions cleared were between the 6 S3 satellites, if such an HIE were to be screened, the MSMA would choose the highest system-wide reward across the maneuver tradespace for *both* spacecraft, i.e. another stack of 30 circles for that conjunction in Fig. 6-

top, and another 30 markers of the same color but different shape in Fig. 6-bottom. Only one satellite with the higher reward would maneuver. This behaviour has been verified on single events, however it is difficult to replicate in a cascading conjunction scenario unless it is the first encountered HIE (clearing one HIE often clears other HIEs between two target satellites). The output in either scenario shows the temporal order of maneuvers to clear conjunctions over the planning period, where the MSMA successfully chooses the largest possible reward to clear each conjunction in a chronological sequence. This recommendation is then presented to the S3 for selection and then to the operator for execution via the STM framework.

Per-conjunction reward maximization via MSMA does not necessarily lead to global optima (global maximum reward), but it is significantly more tractable and provides a reasonable baseline against which other more sophisticated solutions can be compared. More importantly, this simple example demonstrates the need of automated and distributed planning in a crowded future space environment where sequential conjunctions and cascading consequences of maneuvers to avoid them will be common. The STM architecture is expected to spur research in these areas, and the STM prototype is expected to enable testing them.

3.4. Maneuver negotiation

The planning example we presented involved a conjunction between multiple maneuverable satellites, all controlled by the same S3. If multiple maneuverable satellites within a CDM are controlled by different S3s, maneuver negotiation may be needed before accepting and publishing a maneuver. Currently in the absence of STM, such negotiations are rare and occur by email, telephone, or fax (if at all). More commonly, an operator simply informs CSpOC of an intended maneuver, evaluates screening results, and executes the maneuver without negotiation. This process has functioned thus far because conjunctions are rare—most conjunctions involve only one maneuverable satellite (the other being propulsion-less, e.g. CubeSat or debris)—and the overhead of negotiation and legal implications has been perceived to outweigh the benefits. For example, the SDA process has successfully facilitated the negotiation that our MSMA system simulates, whereby multiple maneuverable spacecraft are coordinated by different S3s, namely the SDC and the 18SPCS system.

As conjunctions become more common and maneuvers result in cascading secondary or tertiary conjunctions, maneuver negotiation is envisioned when CDMs involve single or multiple maneuverable satellites by different S3s. Consensus may be established by S3s communicating through the STM network and APIs, or a brokering service that is an SDS within system. A future version of the STM prototype will define such APIs and implement example S3s (and brokers) to demonstrate the utility of such standardization.

Who maneuvers may be decided by rule-based systems like those used by sailboats, Coasian-based payments based systems where one would pay the other to maneuver and determine a price cheaper than its own cost to maneuver (e.g., a currency quantification of Equation (1)), auction-based systems where such payment prices are bid and counterbid between the negotiators, resource-based systems which choose the higher reward or lower cost operator (e.g., evaluated using Equation (1)), dual-maneuver implicit cost split where both maneuver with equitable cost (e.g., by optimizing the total reward across both operator satellites using Equation (1)), or “space chicken” where it is assumed that one operator will eventually decide to move to avoid an imminent HIE. These are discussed in detail in Ref. [14]. Modifications to quantitatively account for practical preferences such as fuel weight in reward, combining COLA maneuvers with pre-planned station-keeping or

trusting the more experienced operator with maneuvers are possible for all proposed systems above. Research on such topics will need to take into account rules for public sharing of information, disparity in reward definitions or weights across operators, commercial incentive to over-report mission impact or under-report efficient maneuvers to avoid action, fairness over time so that the onus of action is equitable (an operator is in some way credited for previous maneuvers when considering the next required maneuver), etc.

4. Conclusions and future work

The STM research platform at NASA Ames has workstations, servers and a hyperwall, with NASA and AGI software suites, and is leveraging the UTM experience and codebase for STM development. Fig. 7 shows our development process from left to right: we have proposed an STM architecture in the past, summarized in Fig. 1 [14,15]. The current paper describes the software prototype we have developed using example service providers (e.g. CAS, AMA, SSA), and in-house sample models [17] or publicly available algorithms for internal functions of the services. We are now involving early partners in industry, academia, and government who will be potential operators and higher fidelity service providers, i.e. potential customers of the STM network. The modular, containerized architecture ensures that they will interact only through APIs and data models, without having to share any proprietary internals of their software. The presented prototype of the STM architecture is for an initial version of TCL1 (on-orbit operations with civil catalogs). The architecture and prototype will be matured over the TCLs defined in Ref. [15] Fig. 8, and as we improve our use case portfolio and increase partners. The process in Fig. 7 is expected for every TCL.

The STM architecture and standardization of interaction between entities paves the way for a research ecosystem similar to other AI/autonomy fields, such as UAS or self-driving cars. Some planning and scheduling scenarios are listed below in terms of increasing complexity:

1. Single maneuverable satellite to avoid a conjunction with an uncooperative target
2. Single maneuverable satellite to avoid conjunction with a single maneuverable satellite, both controlled by the same S3
3. Multiple maneuverable satellites to avoid sequential conjunctions with multiple maneuverable satellites, all controlled by the same S3
4. Multiple maneuverable satellites to avoid sequential conjunctions with multiple maneuverable satellites, controlled by different S3s

We applied the STM prototype to a multi-satellite scenario that required sequential COLA actions to prevent frequent ‘cascading’ conjunctions (#3). The scheduling algorithm in our example AMA is capable of handling scenarios #1, #2 and #3, and we discuss the negotiation implications required to extend to #4. We also presented an example reward function and an example planning algorithm to advise on the schedule of COLA maneuvers across multiple satellites. Simple examples in this paper are expected to serve as a strawman for future, high fidelity STM services acknowledging that there may be uncertainties in the quantification of risk and reward models given the international complexity of the STM environment; e.g. COLA between government/military satellites of a different country whose operator is unknown. While this paper focused on CAS and COLA related applications of autonomy within the STM, there is future scope for AI/autonomy development for other aspects of STM such as validating and merging SSA catalogs (e.g. with SDC APIs), decision-making as a function of uncertainty in data unavailability, and COLA as a function of varying levels of data sharing, as discussed in detail in Ref. [14].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project was sponsored by NASA Aeronautics Research Mission Directorate. The authors would like to thank Dr. Joseph Rios and Priya Venkatesh at NASA ARC, and Prof. Mykel Kochenderfer and his student, Nolan Johnson, at Stanford University for useful discussions on the topics presented.

Appendix A. Benefits of MicroServices

We used a micro-service architecture to deploy the STM prototype owing to the following major benefits [24,25]:

- Complexity localization — Services are self-contained, independent applications. The development team for each service (S3, CAS, etc.) is only concerned with understanding the complexities of their service. Other teams only need to know what capabilities are being provided by the other services; they don't need to know how they work internally.
- Cross-cutting business functionality — Eliminates the need to reinvent standard pieces of functionality used across the organization multiple times; for example, authentication and user management.
- Increased resiliency — Since a number of services communicate simultaneously, when one fails, the client should be designed to allow its neighbors to continue functioning as it steps out as elegantly as possible. Improved fault isolation means an uninterrupted user experience.
- Better scaling, efficient system optimization and organization — Scaling decisions can be made at a more granular level.
- Output flexibility — Simplified data extraction.
- Real-time processing support — The publish-subscribe framework facilitates data processing in real time.
- Support for best technology selection — One is not limited to a single technology set for the overall project. Each microservice can be developed using the most appropriate programming language and data storage technology for its function.
- Scalability — High level of code and data reuse, making it faster and easier to deploy additional services to address new use cases.
- Security flexibility — Allows applications to segment off and outsource their non-core business functions without disclosing core services.
- Experimentation flexibility — Ability to try out a new technology stack on an individual service. Compared to monolithic designs, any dependency concerns will be smaller and rolling back changes is simpler. It also eliminates any long-term commitment to a single technology stack.
- Coordination — Uses event-streaming technologies to enable easy integration compared to the heavyweight inter-process communications protocols.

Appendix B. Technology Comparison between Software Stack Components

The web frameworks considered for User Interfaces were:

- Angular 2+: Popular framework, large support community, MIT license, maintained by Google.
- React: Supported by Facebook, large developer support community.
- Django: Python web-framework, high scalability, offers high security, provides rapid development.

Angular 5 was chosen as the web development framework due to its small package size, extensibility, support of simple progressive web application, material design, in-build code optimization, extensive capabilities to interface with APIs and our familiarity with it.

The Application Programming Interface frameworks considered were:

- Node. JS: Javascript-based runtime environment providing non-blocking, event driven servers due to its single-threaded nature; remains lightweight and efficient in the face of data-intensive real time applications that run across distributed devices.
- Flask: Python-based lightweight web application framework designed to be quick and easy, with the ability to scale up to complex applications.

Node.JS was chosen as the preferred API framework.

The Database tools considered were:

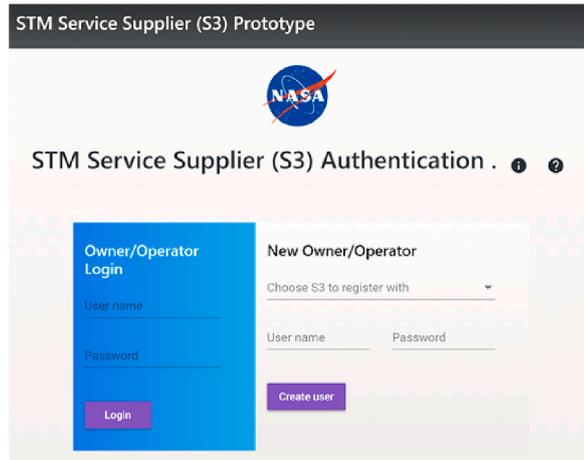
- MongoDB: Open source document-based database management tool that stores data in JSON formats; Highly scalable, flexible and distributed NoSQL database; Schema-less, no complex joins, ease of scale out, conversion/mapping of application objects to database objects not needed.
- Couch-base: Open-source NoSQL, multi-model, document oriented DB management system that store JSON documents.
- PostgreSQL: Object oriented relational DB with an emphasis on extensibility and standards compliance; functions and operators can be used with JSON and JSONB.

MongoDB was chosen due to its Docker friendly images, ability to store JSON, capability to change streams which enables pushing updates across and from various service providers and STM roles.

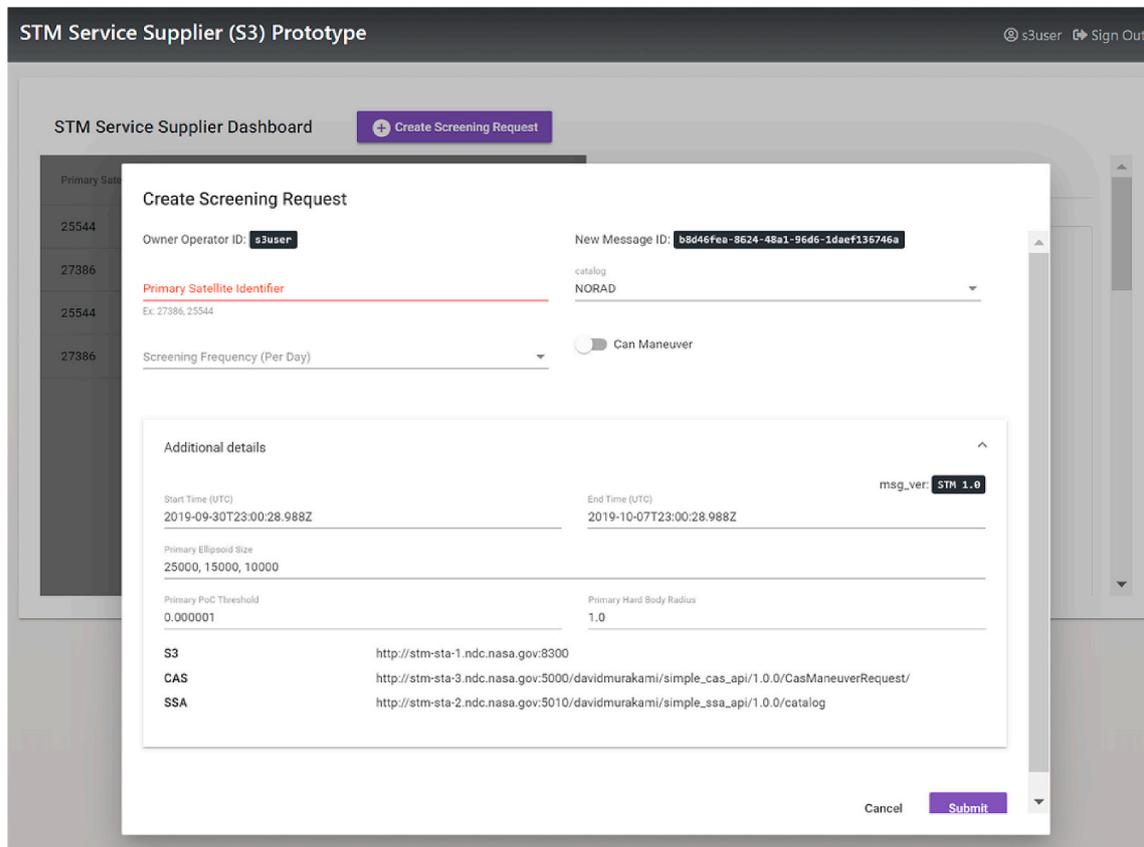
Appendix C. Screenshots from the STM software prototype

This section shows some screenshots from the execution of the software prototype, as executed end to end for the case study presented in Section 3. The description of the APIs and the sequence narrative is available in Section 2.3.

1. Login and registration page for any S3, owners or operators



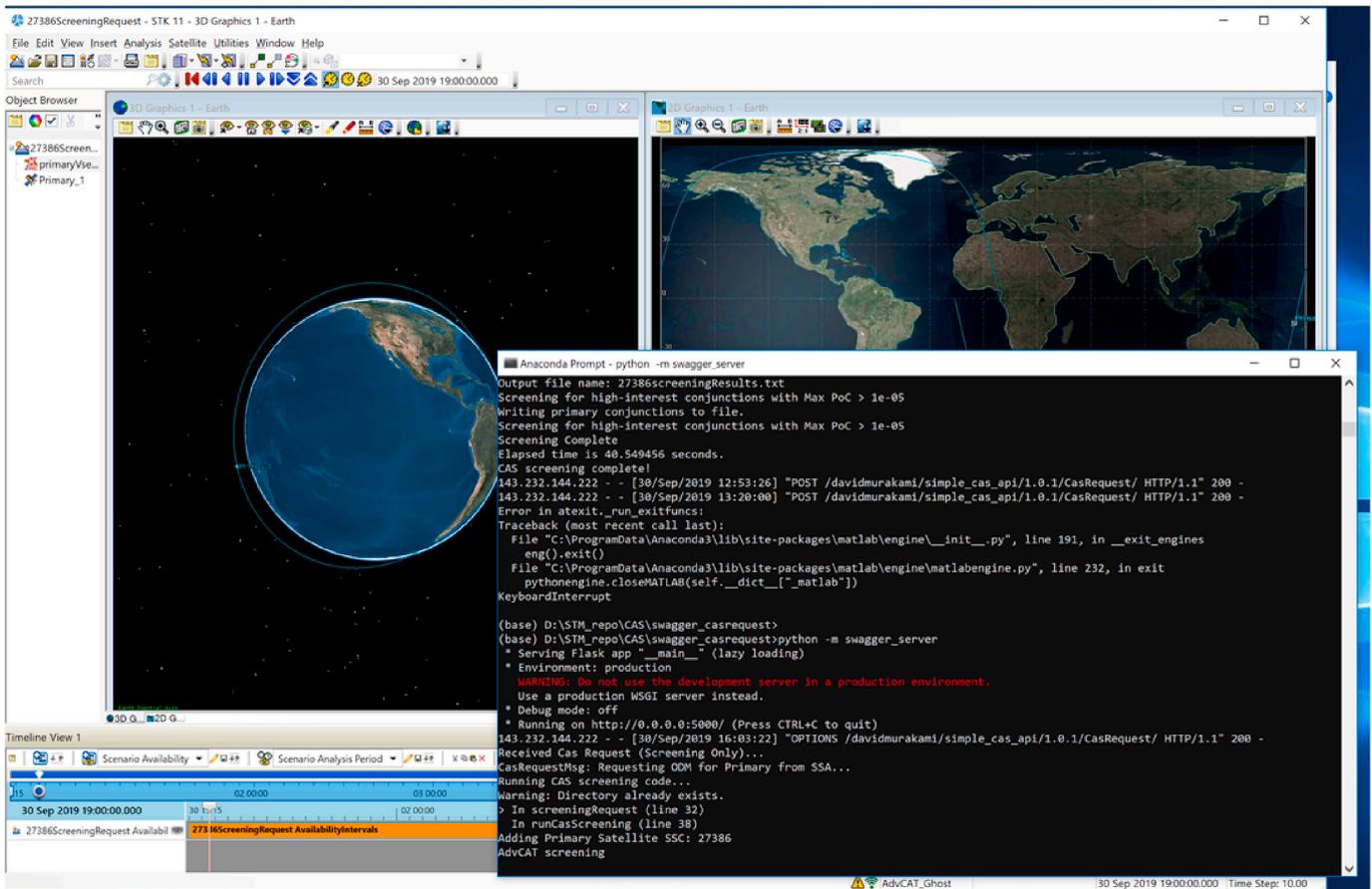
2. Screening request UI for an O/O spacecraft



3. SSA server log showing the receipt of the S3 request

```
(base) D:\STM\SSA\swagger_ssarequest>conda activate SSAEnv2
D:\STM\SSA\swagger_ssarequest>set "JAVA_HOME_CONDA_BACKUP="
D:\STM\SSA\swagger_ssarequest>set "JAVA_HOME=C:\ProgramData\Anaconda3\envs\SSAEnv2\Library"
(SSAEnv2) D:\STM\SSA\swagger_ssarequest>python -m swagger_server
CONTROLLER IMPORTS SUCCESSFULL
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:6500/ (Press CTRL+C to quit)
Grabbing TLE from Space Track...
PROPAGATING THE FOLLOWING TLE:
0 ENVISAT
1 27386U 02009A 19273.63596506 .00000006 00000-0 15578-4 0 9995
2 27386 98.1463 289.3283 0001186 89.5620 18.7558 14.37958425921138
LINE 1:
1 27386U 02009A 19273.63596506 .00000006 00000-0 15578-4 0 9995
1 27386U 02009A 19273.63596506 .00000006 00000-0 15578-4 0 9995
RESULTANT STATE VECTORS:
{-1,649,965.4246762183; 1,802,698.2116883658; 6,704,289.282157258}
{-2,034.0315932909; 6,804.0700887374; -2,325.2021426674}
143.232.144.251 - - [30/Sep/2019 16:11:39] "POST /davidmurakami/simple_ssa_api/1.0.0/catalog HTTP/1.1" 200 -
```

4. CAS server initiates the screening process on proprietary software



5. API to return the CDMs to the S3 after screening results have been processed by the CAS

The screenshot displays the STM Service Supplier (S3) Prototype interface. At the top, there is a header with the user 's3user' and a 'Create Screening Request' button. Below this is the 'STM Service Supplier Dashboard' with a table listing satellites and their screening requests. A modal window titled 'Cas Request Header' is open, showing details for a screening request. The modal contains two identical sections of data, each starting with a checked checkbox 'Select NORAD 27386 for maneuver advisor'. Each section lists Primary and Secondary RSO (Right of Space Operations) and ODM (Orbit Determination Method) parameters. The Primary RSO parameters include Catalog (NORAD), Identifier (27386), Epoch, Frame (Earth Inertial), Position, and Velocity. The Secondary RSO parameters include Catalog (NORAD), Identifier (32316), TCA, MaxPoC, and RangeTCA. The Primary ODM parameters include Epoch, Frame (Earth Inertial), Position, and Velocity. The Secondary ODM parameters include Epoch, Frame (Earth Inertial), Position, and Velocity. At the bottom of the modal, there is a 'Get maneuvers' button and a 'Close' button.

Primary Satellite	Originating User	Creation Time	Screening request	Screening response
25544	s3user	2019-09-30T1	msg_ver	STM 1.0
27386				
25544				
27386				
27386				
27386				
27386				
27386				

Screening Request..

Velocity:
0.19686975047327832,-6.414916383938387,3.8394429931988623

Select NORAD 27386 for maneuver advisor

Primary RSO
Catalog: NORAD
Identifier: 27386

Secondary RSO
Catalog: NORAD
Identifier: 39783
TCA: 2019-10-05T22:54:40.677753735
MaxPoC: 0.000012541463466596509
RangeTCA: 2.4863116434141214

Primary ODM
Epoch: 2019-10-05T22:54:40.677753735
Frame: Earth Inertial
Position:
675.9714180635317,956.6253028402267,-7055.037296254703
Velocity:
3.1527637078401956,-6.728067750312608,-0.6105107972232635

Secondary ODM
Epoch: 2019-10-05T22:54:40.677753735
Frame: Earth Inertial
Position:
675.9897411461338,956.2406886740616,-7057.49361113487
Velocity:
-6.379643791794466,-3.67508964732553,-1.1602376677069224

Select NORAD 27386 for maneuver advisor

Primary RSO
Catalog: NORAD
Identifier: 27386

Secondary RSO
Catalog: NORAD
Identifier: 32316
TCA: 2019-10-06T06:24:07.160635053
MaxPoC: 0.00001772981839262378
RangeTCA: 2.05788402678478

Primary ODM
Epoch: 2019-10-06T06:24:07.160635053
Frame: Earth Inertial
Position:
-386.22341359285946,-1544.6158542385715,6955.193550705665
Velocity:
-3.246239677686192,6.606934291394185,1.2846268077600385

Secondary ODM
Epoch: 2019-10-06T06:24:07.160635053
Frame: Earth Inertial
Position:
-385.341123855458,-1545.3187158224737,6956.914723875583
Velocity:
-6.583561176216058,-3.380254601197688,-1.0825750879590117

Get maneuvers

Close

Appendix D. MSMA Maneuver Details for Sequential Conjunctions

Listed below are the details of maneuvers recommended by the MSMA (assumed selected and executed) for 16+ conjunctions in the COLA case study of Fig. 6:

Man. Num	Maneuver Time	TCA	Primary	Time (Epoch Hours)	Color
1	2020-05-20T19:54:36.070722	2020-05-20T23:33:53.754326020	25419_5	4.56	
2	2020-05-20T20:26:44.711755	2020-05-21T01:13:50.541275825	42809_2	6.23	
3	2020-05-20T22:52:59.257982	2020-05-21T20:27:37.818446621	42961_3	25.46	
4	2020-05-21T10:30:17.590531	2020-05-21T18:16:34.607796485	42961_3	23.28	
5	2020-05-21T20:04:39.135977	2020-05-22T02:27:00.576795342	42961_3	31.45	
6	2020-05-22T01:21:11.447624	2020-05-22T09:16:59.578368321	42961_3	38.28	
7	2020-05-22T05:32:13.105643	2020-05-22T11:48:06.839776321	10676_0	40.8	
8	2020-05-22T06:59:13.128231	2020-05-22T12:49:52.716927420	25419_5	41.83	
9	2020-05-22T15:09:04.238103	2020-05-23T10:05:56.869794060	42809_2	63.1	
10	2020-05-23T03:15:58.770464	2020-05-23T15:24:52.333031771	42961_3	68.41	
11	2020-05-23T05:00:31.153382	2020-05-23T20:50:15.276143643	25477_4	73.84	
12	2020-05-23T13:09:10.848066	2020-05-24T01:19:35.631353521	41918_1	78.33	
13	2020-05-23T20:26:33.955465	2020-05-24T03:42:54.539739549	42809_2	80.72	
14	2020-05-26T05:30:01.905980	2020-05-26T11:50:00.577205342	41918_1	136.83	
15	2020-05-26T17:03:37.196967	2020-05-26T22:00:27.768348550	41918_1	147.01	
16	2020-05-27T04:56:01.138035	2020-05-27T06:15:15.707116806	41918_1	155.25	

References

- [1] W. Ailor, G. Peterson, J. Womack, M. Youngs, Effect of large constellations on lifetime of satellites in low earth orbits, *J. Space Saf. Eng.* 4 (3) (Sep. 2017) 117–123, <https://doi.org/10.1016/j.jsse.2017.11.003>.
- [2] G. Peterson, M. Sorge, W. Ailor, New Space Activities and Implications for Space Traffic Management and Conjunction Assessment, Aerospace Corporation, 2018.
- [3] S. Alfano, D.L. Oltrogge, R. Shepperd, LEO Constellation Encounter and Collision Rate Estimation: an Update, 2020.
- [4] D. McKnight, M. Matney, K. Walbert, S. Behrend, P. Casey, S. Speaks, Preliminary Analysis of Two Years of the Massive Collision Monitoring Activity, 2017.
- [5] J.A. Haimerl, G.P. Fonder, Space fence system overview, in: Presented at the Advanced Maui Optical and Space Surveillance Technology Conference, Maui, Hawaii, USA, 2015.
- [6] G.P. Fonder, P.J. Hack, M.R. Hughes, AN/FSY-3 Space Fence System-Sensor Site One/Operations Center Integration Status and Sensor Site Two Planned Capability, 2017.
- [7] C. Contant-Jorgenson, P. Lála, K.-U. Schrogl, The IAA cosmic study on space traffic management, *Space Pol.* 22 (4) (2006) 283–288.
- [8] K. Bilimoria, R. Krieger, Slot architecture for separating satellites in sun-synchronous orbits, in: AIAA SPACE 2011 Conference & Exposition, 2011, p. 7184.
- [9] C.R. Phipps, C. Bonnal, A spaceborne, pulsed UV laser system for re-entering or nudging LEO debris, and re-orbiting GEO debris, *Acta Astronaut.* 118 (2016) 224–236.

- [10] J. Mason, J. Stupl, W. Marshall, C. Levit, "Orbital debris–debris collision avoidance, *Adv. Space Res.* 48 (10) (2011) 1643–1655.
- [11] T.J. Muelhaupt, M.E. Sorge, J. Morin, R.S. Wilson, Space traffic management in the new space era, *J. Space Saf. Eng.* 6 (2) (2019) 80–87.
- [12] M.A. Skinner, M.K. Jah, D. McKnight, D. Howard, D. Murakami, K.-U. Schrogel, Results of the international association for the advancement of space safety space traffic management working group, *J. Space Saf. Eng.* 6 (2) (2019) 88–91.
- [13] D.L. Oltrogge, I.A. Christensen, Space governance in the new space era, *J. Space Saf. Eng.* 7 (3) (2020) 432–438.
- [14] S. Nag, D. Murakami, M. Lifson, P. Kopardekar, System Autonomy for space traffic management, in: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference, DASC, London, United Kingdom, 2018, pp. 1–10.
- [15] D.D. Murakami, S. Nag, M. Lifson, P.H. Kopardekar, Space Traffic Management with a NASA UAS Traffic Management (UTM) Inspired Architecture, 2019. San Diego CA.
- [16] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, J. Robinson, Unmanned aircraft system traffic management (utm) concept of operations, Accessed: Feb. 14, 2017. [Online]. Available: https://www.researchgate.net/profile/Joseph_Rios2/publication/303902685_UAS_Traffic_Management_UTM_Concept_of_Operations_to_Safely_Enable_Low_Altitude_Flight_Operations/links/57e12df408ae9834b4e7e459.pdf, 2016.
- [17] J.V. Cabrera, S. Nag, D.D. Murakami, An initial analysis of automating conjunction assessment and collision avoidance planning in space traffic management, in: Presented at the AAAS Spaceflight Mechanics Conference, Maui, Hawaii, USA, 2019.
- [18] M.A. Skinner, Conceptual development of a civil space traffic management system capability, in: Advanced Maui Optical and Space Surveillance Technologies Conference, 2018, pp. 11–14.
- [19] T.S. Kelso, Analysis of the Iridium 33-Cosmos 2251 Collision, Analutical Graphics Inc., Feb. 2009 [Online]. Available: <https://www.agi.com/resources/whitepapers/analysis-of-the-iridium-33-cosmos-251-collision>.
- [20] S. Alfano, D.L. Oltrogge, H. Krag, K. Merz, R. Hall, Risk assessment of recent high-interest conjunctions, in: Presented at the 71st International Astronautical Congress (IAC), Virtual, Oct. 2020.
- [21] Federal Communications Commission, Space Exploration Holdings, LLC Request for Modification of the Authorization for the SpaceX NGSO Satellite System, Apr. 26, 2019 [Online]. Available: <https://www.fcc.gov/document/international-bureau-grants-spacexs-modification>.
- [22] C. Bombardelli, J. Hernando-Ayuso, Optimal impulsive collision avoidance in low earth orbit, *J. Guid. Contr. Dynam.* 38 (2) (2015) 217–225.
- [23] S. Nag, et al., Autonomous scheduling of agile spacecraft constellations with delay tolerant networking for reactive imaging, in: Presented at the International Conference on Planning and Scheduling (ICAPS) SPARK Applications Workshop, U. S.A., Berkeley, California, 2019.
- [24] J. Thönes, Microservices, *IEEE Softw.* 32 (1) (2015), 116–116.
- [25] N. Dragonì, et al., Microservices: yesterday, today, and tomorrow, in: Present and Ulterior Software Engineering, Springer, 2017, pp. 195–216.