# Visualisation Tools for Satellite Digital Twins
Academic Year 2024/25

Ricardo B. B. Santos[a], Paulo J. S. Gil[b], João P. L. Monteiro[c]

[a]*MEAer, Instituto Superior Técnico, Universidade de Lisboa*
[b]*LAETA, IDMEC, Instituto Superior Técnico, Universidade de Lisboa*
[c]*ISR-Lisboa, Instituto Superior Técnico, Universidade de Lisboa*

**Abstract**

This project presents the design and development of a web-based visualization platform for satellite Digital Twins (DTs), emphasizing user interaction and visual clarity over data modeling. While existing efforts in space DTs focus primarily on simulation fidelity and fault diagnostics, the visualization layer remains underdeveloped despite its critical role in operator comprehension and decision-making. This work addresses that gap by proposing a modular, open-source system capable of integrating real-time telemetry, orbital and attitude data, and environmental parameters into a coherent interface.

The system follows a client-server architecture, with a React and React Three Fiber frontend for rendering 2D and 3D views, and a FastAPI backend using GODOT for orbital prediction. Key features include 2D ground-track visualizations, 3D orbit and attitude displays, embedded telemetry dashboards, and support for multiple satellites. Requirements were defined through literature review, discussions with the ISTSAT team, and interviews with satellite operators, ensuring practical relevance. Operators prioritized ground-track and attitude visualizations, which guided development choices.

A major architectural decision was to treat the DT as observational, avoiding direct control interventions and focusing solely on visualization. The communication layer relies on an intelligent polling REST API, with extensibility toward alternative protocols. Current progress includes multi-satellite orbit tracking from TLE inputs, temporal navigation, and foundational support for additional visualization layers.

This platform contributes a missing layer in the DT ecosystem by offering a scalable, user-centered approach to interacting with satellite data. It establishes a foundation for future expansion toward interactive or mission-specific DT interfaces and supports the broader adoption of DTs in space operations.

*Keywords:* digital twin, satellite systems, visualization tools, orbital trajectories, web-based interfaces

*Email address:* `ricardo.belchior@tecnico.ulisboa.pt` (Ricardo B. B. Santos)

## Contents

## 1. Objectives and Motivation

This work develops visualization tools for satellite Digital Twins (DTs), ensuring that the representation of telemetry data, fault diagnostics, and predictive analytics is both accessible and actionable. Unlike previous studies that emphasize the development of DTs themselves, this work prioritizes the user interface and human-computer interaction aspects, aiming to create tools that bridge the gap between raw data and operational insights.

The increasing complexity of satellite systems, coupled with the vast amounts of data generated by onboard sensors, has led to significant challenges in monitoring, analyzing, and interacting with these systems in real time. DTs have emerged as an ambitious and transformative technology for space applications, offering a virtual counterpart to physical satellites that allows real-time simulation, fault diagnosis, and predictive maintenance [1]. However, effective visualization tools are crucial to unlocking the full potential of DTs, as they provide intuitive ways for operators to interact with complex datasets and enhance decision-making [2].

This work aligns closely with the Neuraspace “AI Fights Space Debris” initiative, which aims to enhance space traffic management through AI-powered collision prediction and avoidance [3]. By situating our visualization platform within this broader context, we provide a critical human-facing layer: intuitive tools for interpreting AI-generated conjunction alerts, planning avoidance maneuvers, and monitoring debris proximity in real time. This integration ensures that complex backend analytics are not just accurate but also readily actionable by operators.

## 2. Visualization and DTs

In this section, a brief history of both visualization techniques and the DT concept are presented. Some important distinctions are also made between different types of DTs.

Visualization in software has deep historical roots, with early examples dating back to the 2nd century AD when drawings and visual representations were used to investigate natural phenomena and record historical events [4]. The field saw significant advances in the 17th to 19th centuries, such as the creation of thematic maps and the invention of bar and line charts, which enabled more systematic representation of statistical data. The so-called ”Golden Age” of statistical graphics, roughly from 1840 to 1910, was marked by elegant and innovative visualizations that profoundly shaped how data was communicated and understood [5]. The emergence of computer technology in the late 20th century revolutionized visualization, making it possible to manipulate and explore large datasets interactively and efficiently [4].

The importance of visualization in software lies in its ability to transform raw data into meaningful insights, making complex information accessible and actionable for a wide range of users [4]. Effective visualizations help users quickly identify trends, patterns, and anomalies, supporting informed decision-making in fields as diverse as business, healthcare, and scientific research [6]. As data volumes have grown, visualization has become indispensable for both analysis and communication, allowing users to distill key messages from intricate datasets and present them in a visually compelling manner [4]. The rise of interactive and dynamic visualizations has further enhanced users' ability to explore data, test hypotheses, and share findings with diverse audiences [6].

Common categories of visualization in software include basic charts such as bar, line, and pie charts; statistical plots like scatter plots and histograms; and more advanced forms such as heat maps, tree maps, and network diagrams [6]. Geographic maps are widely used for spatial data, while dashboards integrate multiple visualizations for real-time monitoring and analysis. Interactive visualizations, which allow users to filter, zoom, and manipulate data, have become increasingly prevalent with the advancement of web technologies and specialized software tools. The choice of visualization type depends on the nature of the data, the analytical goals, and the intended audience, underscoring the importance of thoughtful design in effective data communication [6].

The concept of 'twin' arises in the NASA Apollo program, where duplicates of the space vehicles are built in order to subject them to the same conditions that these would experience in their missions [7]. Then with the development of digital technology and simulation capabilities it became increasingly possible to encapsulate aspects of a physical entity within a digital environment [1]. Being originally intended for the Fault Diagnosis and Health Monitoring of aircraft [8], the concept of DT has since been successfully applied to other mechatronic systems [9–11], including space systems.

A DT is a detailed virtual model that replicates the behavior and performance of a physical system. It combines advanced simulations with real-time data and historical records to accurately represent key components, such as the structure, propulsion, energy systems, and avionics, providing a realistic and dynamic counterpart to the actual system [8].

DTs for on-orbit spacecraft present unique demands. In typical industrial or terrestrial contexts, DTs are used to support maintenance, performance monitoring, and optimization, often within well-connected and accessible environments. Unlike terrestrial systems, spacecraft operate in remote and inaccessible conditions, making the DT not just a support tool but a critical element for autonomous monitoring and decision-making. A space DT must reflect the condition of the satellite in real time, accounting for limited communication bandwidth, latency, and a hostile external environment.

These twins integrate telemetry and environmental data to simulate system behavior under dynamic conditions, enabling fault detection, performance prediction, and mission adaptation without physical intervention. They also support cooperative satellite operations, where multiple DTs share data and optimize behavior collectively.[12]

It is also important to introduce the concept of data-driven DT (DDDT). This approach shifts the paradigm by relying not on intrinsic physical knowledge but on the observed behavior of the system, collected from a variety of data sources. Instead of constructing simulations from internal design documentation, the DDDT learns patterns and relationships directly from telemetry, environmental data, and ground observations. This makes it possible to develop DTs even when detailed schematics or design intent are missing—an especially valuable capability in the satellite domain, where many spacecraft lack publicly available or complete technical information [13].

An important distinction between DTs is whether the DT software allows for direct control interventions. Hence, the mutually exclusive concepts of Observational DTs (ODTs) and Interactive DTs (IDTs) are introduced in this work. ODTs are a type of DT that focus solely on monitoring and analysis, providing a detailed virtual representation of the state and behavior of the satellite without allowing direct control interventions. This approach maintains a clear separation between monitoring and control systems, often preferred for safety-critical space operations where control actions require separate validation and authorization protocols [14]. One such example of a ODT is given in [15] outside the space domain. In contrast, Interactive DTs (IDTs) extend beyond monitoring to include command and control capabilities, allowing operators to directly influence the behavior of the physical-entity through the digital interface. While this integrated approach can streamline operations, it requires additional safety measures and validation mechanisms to prevent unintended consequences in the physical system. One such example of an IDT is given in [16].

## 3. State of the Art

In this section, the state of the art DT technologies and visualization softwares for space systems is presented. First, the range of applications of the DT concept is presented, followed by a more detailed discussion on its application to space objects. Then, different existing visualization tools for satellite systems are presented and discussed, leading to the identification of the need for new tools applicable to satellite DTs.

### *3.1. DTs and Space DTs*

Since 2017, the concept of DT has gained increasing attention in research, leading to its application across a wide range of fields [17]. In the energy sector, DTs have been applied to wind farms to improve operational efficiency and predictive maintenance [18]. In healthcare, they have been explored for personalized treatment planning and monitoring [19]. Smart cities have adopted DT frameworks to simulate urban infrastructure and optimize resource management [20]. In construction, DTs have been used to model building behavior and improve project coordination [21]. Additionally, the concept of a DT shop-floor (DTS) has been proposed to support intelligent design and optimization [17].

In 2012, the DT was proposed as a high-fidelity, real-time simulation designed to mirror the lifecycle of an aerospace system, aiming to overcome the limitations of heuristic-based certification and maintenance [8]. The evolution and future potential of the Spacecraft DT have been outlined, emphasizing its role in enabling autonomous cognition, operations, and cluster collaboration by leveraging real-time data integration, modeling, and cloud-based services [12]. A comprehensive DT-based framework is proposed for satellite fault diagnosis and health monitoring, leveraging real-time synchronization between physical systems and multi-domain virtual models to outperform traditional data-driven methods in accuracy, reliability, and interpretability [1]. A different angle is offered, that emphasizes the use of DTs during early concept design phases, proposing a modular, synchronized simulation framework that integrates with system models and authoritative data sources to validate spacecraft concepts before physical realization [22].

A case study demonstrates a DT framework for reusable spacecraft, combining model-based simulation and dynamic Bayesian networks to track fatigue crack growth and improve reusability evaluation through continuous uncertainty reduction and real-time updates [23]. A detailed DT system for spacecraft assembly is developed, integrating real-time data acquisition, event-driven simulation, and multi-layer synchronization across physical and virtual domains to enhance traceability, process monitoring, and quality control in manually intensive assembly environments [24]. Significant advances are made through the development of a mechanism-data-driven DT platform for spacecraft, integrating hybrid modeling, real-time monitoring, and virtual-real interaction to support critical on-orbit operations such as docking, orbit entry, and astronaut activity simulation [25].

A technically detailed review of space environment DTs emphasizes their role in modeling complex orbital dynamics, radiation effects, and debris interactions to improve mission planning, risk assessment, and adaptive control strategies [26]. A low-cost, flexible DT platform is described that enables real-time degradation assessment of spacecraft lithium-ion battery packs using algorithmic modules to estimate State of Charge (SOC) and Remaining Useful Life (RUL) based on sensor data and predictive modeling [27]. A formal methodology is introduced for constructing spacecraft operations DTs using fault and behavior trees, enabling real-time fault detection, isolation, and mitigation through executable, operator-oriented models derived from system reliability data [28]. A DT framework grounded in space informatics is developed to enhance spacecraft resilience, integrating artificial intelligence techniques to optimize fault management, mission reconfiguration, and dynamic adaptation under uncertain space conditions [29].

Despite these advances, the topic of visualization—particularly how DTs convey state, behavior, and predictions to human users—remains underdeveloped. Most studies focus on model accuracy, data integration, or algorithmic sophistication, but offer limited detail on how complex information is visually presented and interacted with. While some works implicitly highlight the importance of visualization in early design validation [22] or suggest the need for layered visual interfaces through multi-layer synchronization [24], explicit visualization frameworks remain largely absent in the reviewed literature, revealing a clear research gap.

Given the increasing complexity of spacecraft systems and the growing reliance on DTs for critical decisions, effective visualization becomes essential. It serves not only as a diagnostic and monitoring aid but also as a means of building trust in model predictions and enabling collaboration between human operators and autonomous systems. Addressing this gap in the thesis could provide a significant contribution to the field, particularly in developing visualization tools that are scalable, intuitive, and tightly integrated with real-time DT platforms.

## *3.2. Visualization Tools for Satellite Systems*

There is a variety of space related software with overlapping features. This analysis includes software that is used for satellite systems and excludes most software that is used for more general space missions.

The information that can be visualized in a satellite system includes 3D orbit visualization for spatial awareness of orbital paths [30], 3D attitude visualization to depict the orientation of the satellite in space [31], and 2D attitude visualization for simpler reference views [32]. A cartographic 2D orbit visualization can also be provided, displaying the ground track of the satellite on an Earth map [33, 34]. The path of the satellite can also be visualized in the point of view of the ground station, allowing the operator to verify if an antenna on the ground is pointing in the correct direction during the contact window, this is called Zenith View by some softwares [32]. Furthermore, telemetry parameter visualization can also be implemented using time-series plots of key variables such as battery charge and thermal data, along with cross-parameter analysis through interactive graphing tools [35]. This parameter visualization is often included in the form of a dashboard, which is a common way to visualize data in the space domain. Environmental visualizations, such as ambient magnetic field strength [36], and uncertainty visualizations using ellipsoids to represent orbit determination errors (not a standard feature in most mainstream or open-source packages), are also possibilities that offer a holistic picture of satellite operations and environmental context. Finally, relating to payload data, it is also useful to be able to visualize the packages that are received by the satellite.

Starting by mentioning softwares that allow 3D orbit visualization, one of the most complete softwares is STK (Systems Tool Kit) by Ansys, a powerful aerospace and defense simulation platform for modeling, analyzing, and visualizing complex space missions and systems in realistic physics-based environments [37]. Another popular software that achieves this feature, is GMAT (General Mission Analysis Tool) by NASA, an open-source space mission design and trajectory optimization tool focused on high-precision astrodynamics and spacecraft navigation [30].

For attitude visualization through celestial maps and for zenith views, VTS (Visualization Tool for Space data) is an extensible platform developed by CNES for animating and visualizing satellites in 2D and 3D environments, allowing users to configure and synchronize multiple client applications for comprehensive mission analysis [32].

For 3D attitude visualization, the Satellite Scenario framework in MATLAB Aerospace Toolbox and Blockset enables simulation and visualization of space missions with support for multiple satellites, TLE-based orbits, ground station access analysis, and 2D/3D coverage visualization [31]. The software TRACK by Terma [38] and the software MATA (Mission, Attitude, and Telemetry Analysis) have many of the same features [39]. Also with attitude visualization features, 42 is an open-source spacecraft simulation tool developed by NASA, designed for high-fidelity modeling of spacecraft attitude and orbit dynamics [40].

For 2D groundtrack orbit visualization, Orbitron is a free satellite tracking application developed by Sebastian Stoff, designed primarily for amateur radio operators and satellite observers. It provides real-time and simulated tracking of satellites using Two-Line Element (TLE) data, with a focus on antenna alignment and pass prediction [33].

Similar to Orbitron but open-source is Gpredict, an open-source real-time satellite tracking application designed for amateur radio operators and satellite enthusiasts, offering pass prediction, Doppler shift calculation, and 2D cartographic visualization of satellite orbits and ground station visibility [34]. GODOT (Generic Orbit Determination and Optimisation

Toolkit) is an open-source astrodynamics library developed by the European Space Agency (ESA) for orbit and attitude analysis, mission design, and flight dynamics operations. It is designed as a modular, extensible platform for high-precision simulations and analysis, supporting both C++ and Python interfaces [41].

For the visualization of environmental data, there exists one such example is VirES for Swarm which is a highly interactive web-based platform developed for the ESA Swarm mission that enables users to access, visualize, and analyze satellite measurements and geomagnetic models through intuitive 3D and 2D interfaces, advanced filtering, and publication-ready graphics tools [36].

Table 1 summarizes the features and characteristics of each software. It is evident from the analysis of this table that there is no existent open-source software with graphical user interface that achieves the desired features, which motivates the development of visualization software which will be the core of this thesis.

| | GMAT | 42 | GODOT | GPredict | Orbitron | Matlab/ TRACK/ MATA/ STK/ VTS |
|---|---|---|---|---|---|---|
| 3D Orbit Visualisation | ✓ | ✓ | ✓ | | | ✓ |
| 2D Cartographic Orbit Visualisation | | | ✓ | ✓ | ✓ | ✓ |
| Attitude Visualisation | | ✓ | ✓ | | | ✓ |
| Support Multiple Satellites | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Payload Field of View | | ✓ | | | | ✓ |
| Station Visibility | | | ✓ | ✓ | ✓ | ✓ |
| Graphical User Interface | ✓ | | | ✓ | ✓ | ✓ |
| Open Source | ✓ | ✓ | ✓ | ✓ | | |

Table 1: Features and characteristics achieved by each space software

## 4. Preliminary Work

In this section, the preliminary work is presented. This begins with the description of the discussions, the interviews and the research conducted in order to define the requirements for the visualization system. Following that, the desired visualization features are presented and given different levels of priority. After this, the architecture of the visualization system is explained and the respective choices are justified. Finally, the development that has been achieved so far is showcased.

### *4.1. Process of Definition of Requirements*

First, a set of desired features was proposed considering the needs of the company Neuraspace [3]. This information was obtained through continuous discussion with the ISTSAT team, who colaborates with Neuraspace [42]. Concurrently, a bibliographic research was conducted to find what the state of the art could already achieve, and of that state of the art, what was open source, and possible to be integrated in the software. Finally, two interviews were conducted with satellite operators to get their feedback on the requirements

The interviews were conducted in a short semiformal one-to-one meeting with each of the two operators of the NanoSat. Each lasting approximately 30 minutes, the interview began with a general question about the functions the operator achieved in his current workflow. After this, the interviewee was presented with a list of possible features for a visualization system, and was asked to rate them in terms of their importance. Finally, the interviewee was asked if there was any feature that was not listed that he would like to see in the software to be developed.

The feedback from the interviews began with the clarification that the function of the operators was to monitor the satellite NanoSat and store the data received, and not to control it. It was also concluded that obtaining data in real-time, permanently, from the satellite was not only not necessary but also not feasible, since during the contact window between a satellite and a ground station, the amount of data that can be communicated is limited. Then, after presenting the list of features, the operators reported their evaluation of priorities, which is taken into account in Section 4.2. Finally, the operators presented some software that they already used which included Grafana, PlotJuggler, SatDump, and GQRX [35, 43–45]. The mention of Grafana resulted in the addition of the dashboard feature to the list of desired features, as it was reported to be a useful tool for the operators. A typical Grafana dashboard can be observed in Fig. 1.

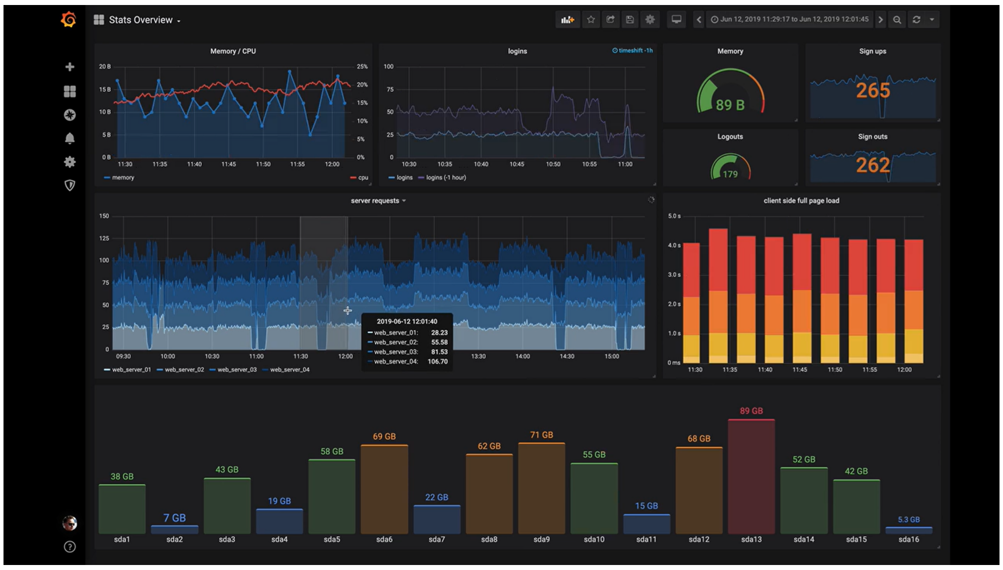

Figure 1: Dashboard Example of Software Grafana

## *4.2. Defined Requirements*

In this part, the desired visualization features are presented and elaborated upon, while being attributed a level of priority in the development. These include orbit visualization, attitude visualization, parameter visualization, dashboard, environmental data visualization, packages visualization, and 3D model visualization.

### *4.2.1. First Order of Priority*

Orbit visualization is a fundamental component of space system monitoring, particularly in the context of DTs, where it provides a spatial understanding of the position and movement over time of the satellite. Some orbit visualizations include a 3D representation of the trajectory of the satellite [30], often relative to the Earth, with support for showing the current orbital path, predicted future trajectory, and past positions. Most systems, however, include a 2D ground track view, which maps the satellite footprint on the surface of the Earth, allowing operators to visualize coverage areas, revisit times, and interactions with ground stations, including the path of the satellite in the point of view of the ground station [33]. Other common features include highlighting key orbital events—such as perigee, apogee, eclipse entry/exit, and maneuver points—and overlaying reference frames like inertial or Earth-centered fixed coordinates. According to the operators interviewed, the 2D ground-track visualization is more useful than the 3D one, hence it will be prioritized.

Attitude visualization complements orbit visualization by representing the satellite orientation in space. Typically, a 3D model of the satellite is displayed, accompanied by vectors that represent its principal axes. In addition, two important reference vectors can also be visualized: one pointing toward the Sun and the other pointing toward the Earth. This 3D view allows operators to easily assess how the satellite is oriented relative to key celestial bodies, which is crucial for tasks such as maintaining communication links, orienting payloads, or protecting sensitive components [31]. For this attitude visualization, an accurate 3D model of the satellite might provide additional information, such as if there are certain sensors which are obstructed from their target by other parts of the satellite. As it is not always guaranteed that a 3D model is available, the attitude visualization should be able to be displayed without it.

In addition to the 3D representation, a 2D cartographic view of the satellite attitude can be provided through a celestial map. In this map, the directions of the satellite principal axes are projected as points onto the celestial sphere, which is then projected onto a 2D plane, giving a graphical indication of the satellite orientation with respect to the stars. The celestial map can also include the plotted positions of specific directions of antennas, sensors, or other critical components. This allows operators to quickly verify whether these components are at risk of undesired exposure, such as pointing toward the Sun (or desired in the case of solar panels), and to plan attitude maneuvers accordingly. Following the feedback of the operators it was concluded that the 2D celestial map could be more useful to measure differences in the directions of certain vectors. This could be achieved, if and only if there was a feature that showed the angle between two selected vectors, since all projection maps distort the distances between points in a spherical surface and distances can't be assessed visually on the 2D map. Hence, it was concluded that the 3D view was complementary to the celestial map, one to get the intuitive understanding of the directions of the vectors and another to have objective measurements of angles between vectors.

*4.2.2. Second Order of Priority*

Beyond orbit and attitude, parameter visualization plays a key role in providing insight into the internal state and health of spacecraft subsystems. Space DTs often track and visualize telemetry related to component temperatures, battery voltages, solar panel outputs, propulsion system performance, and structural loads. Other commonly monitored parameters include reaction wheel speeds, fuel levels, telecommunications link margins, and power subsystem efficiency, all of which contribute to maintaining mission safety and efficiency [13]. The most common method for visualizing these parameters is through time series plots, where the evolution of a variable is displayed over time.

These time series plots are often included in a dashboard, which was considered to be a useful feature by the operators interviewed. As the operators mentioned using Grafana in order to have a dashboard and visualize time series, and as Grafana is a open source tool which allows embedding of dashboards in other applications, it was considered to a good option to integrate it in the visualization system [35]. Additionally, the operators also expressed the need to visualize certain time series superimposed to each other in order to compare them. This could be achieved but is considered of a lower priority since the aim of the visualization software is not to be a time series viewer as complete as PlotJuggler. The minimum requirement is to be able to select points in the time series and to be able to visualize the position and attitude of the satellite at the selected time.

Visualization of environmental data is also an aspect of space DTs, providing operators with essential context about the satellite operating conditions. Environmental data includes information such as solar radiation levels, geomagnetic field variations, atmospheric drag estimates, and charged particle fluxes. These external factors can significantly influence satellite behavior, affecting both orbital dynamics and internal subsystem performance [46]. Visualization tools can map these environmental conditions not only as a whole field around the globe at any given moment in time, but also in relation to the satellite orbit, showing, for instance, periods of higher atmospheric density that may increase drag, or regions with strong radiation belts that could impact electronics. Such representations would allow operators to anticipate operational challenges and plan protective measures or orbital adjustments.

According to the operators interviewed, it could also be useful to visualize the difference between the environmental data considered to be the ground truth and the data measured by the satellite, along its orbit, visualized as a time series. This time series visualization seemed to be more important than a visualization of an entire field around the globe. While this type of visualization could be useful, it was considered to be less important than the other features.

Finally, one of the operators mentioned the need to visualize the packages transmitted by each satellite which should be stored in the DT and to be able to download any sort of data contained in the DT for further analysis.

The conclusion from the interviews, the discussion with the team of ISTSAT and the analysis of the existing literature is that the most important feature to develop is the attitude visualization, followed by the ground-track visualization which would complement it. All other features can be developed later. While the dashboard seems to be the visualization feature most used by the operators, as this is already achieved by Grafana it was considered redundant to develop these same dashboard capabilities, especially when a Grafana dashboard can be embedded in the application.

## *4.3. Implementation*

### *4.3.1. Development Context and Architecture Overview*

A key characteristic of this work is that the visualization system is being developed prior to the existence of a complete DT system. This necessitates the implementation of a backend service to emulate the expected behavior and data output of the DT. Additionally, the communication interfaces must be designed speculatively, anticipating the future needs of DT integration while remaining flexible enough to adapt to changes. This requires careful consideration of data formats and exchange patterns between components. Furthermore, the future DT is assumed to be an Observational DT (ODT), which does not allow for direct control interventions. This separation between monitoring and control systems is not only a good practice in the space domain [14], but also allows for a more focused development of the visualization system, without the need to consider complex control mechanisms.

The satellite visualization system follows a client-server architecture pattern, with a clear separation between the frontend visualization interface and the backend that simulates the behavior of a DT. At the system level, the visualization framework acts as an intermediary between satellite operators and the DT, providing near real-time visual representations of satellite telemetry, orbital parameters, and system states, Fig. 2 illustrates these interactions. The system interfaces with both simulated data during development and is designed to seamlessly transition to real satellite data in production.

As can be seen in Fig. 3, the architecture is organized into three main layers: the React-based frontend for visualization [47], the Python-based backend for simulation and data processing, and a communication layer that facilitates data exchange. This layered approach ensures clear separation of concerns and maintainable code structure.

The architectural patterns employed include the use of React Context for centralized state management, and a RESTful API [48] for all data communication. Because the DT is meant to be observational, the system follows the principle of unidirectional data flow, where data moves from the backend through the communication layer to the frontend contexts, and finally to the visualization components.

The Data Management Layer implements a Transport Interface that abstracts the communication mechanism, allowing the REST API client to be one concrete implementation while maintaining flexibility for future DT developers to implement alternative transport mechanisms without modifying the core application logic [49].

### *4.3.2. Visualization Frontend*

The visualization system employs a component-based architecture, with React as the frontend framework, that divides the frontend into discrete, reusable visualization modules. Each component encapsulates a specific visualization concern—such as orbit display, attitude representation, or telemetry graphing—and consumes the relevant contexts for its data needs. These components are designed to be pure rendering components, focusing solely on

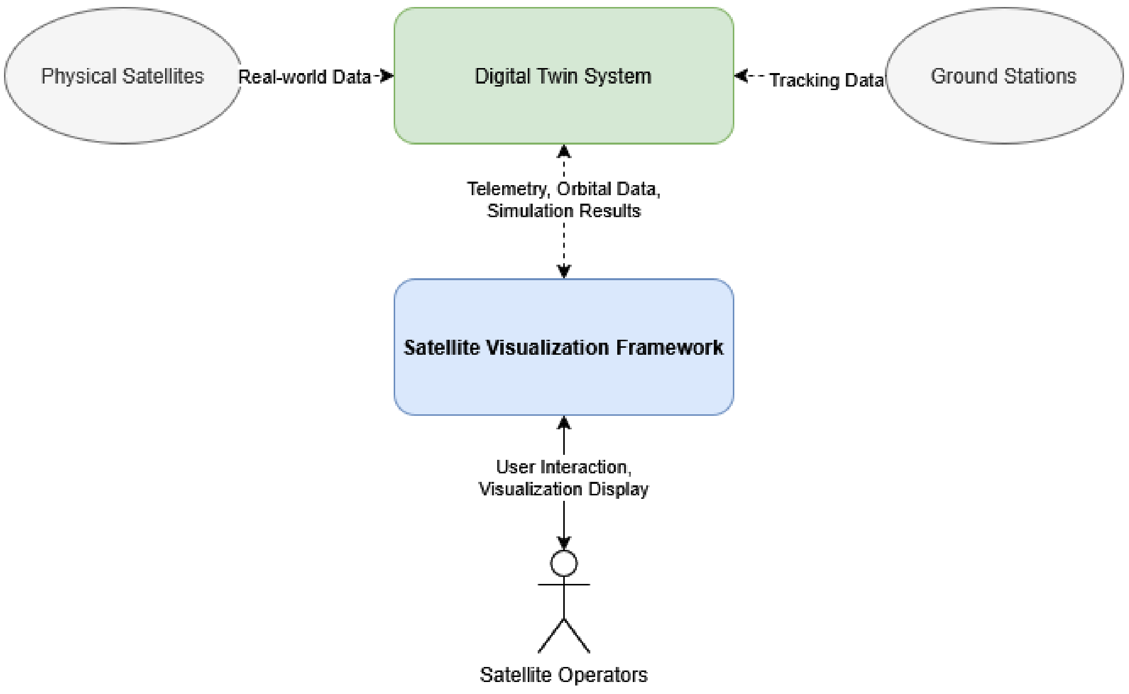
Physical Satellites
Real-world Data
Digital Twin System
Tracking Data
Ground Stations
Telemetry, Orbital Data, Simulation Results
Satellite Visualization Framework
User Interaction, Visualization Display
Satellite Operators

Figure 2: System Overview

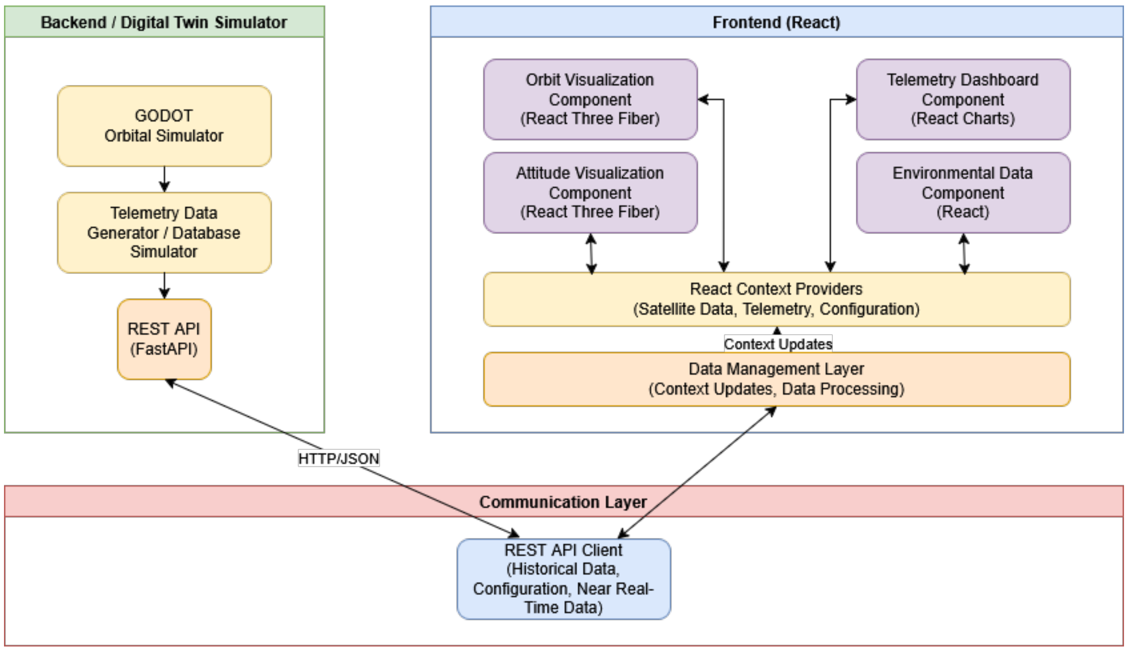
Backend / Digital Twin Simulator
GODOT Orbital Simulator
Telemetry Data Generator / Database Simulator
REST API (FastAPI)
Frontend (React)
Orbit Visualization Component (React Three Fiber)
Telemetry Dashboard Component (React Charts)
Attitude Visualization Component (React Three Fiber)
Environmental Data Component (React)
React Context Providers (Satellite Data, Telemetry, Configuration)
Context Updates
Data Management Layer (Context Updates, Data Processing)
HTTP/JSON
Communication Layer
REST API Client (Historical Data, Configuration, Near Real-Time Data)

Figure 3: Component-Wise Architecture

transforming the data from contexts into visual representations, while remaining decoupled from the complexities of data fetching and state management [47].

The frontend utilizes React Context as its primary state management mechanism, providing a clean and efficient way to share data between components without prop drilling (passing data through intermediate components that don't need it just to get it to deeper components in the React component tree). This architectural choice creates a centralized state management system where visualization components can access shared data through context providers. The Context implementation includes separate contexts for different types of satellite data—such as telemetry, orbital parameters, and system configuration—allowing components to selectively consume only the data they need for their specific visualization tasks. [47].

Meanwhile, the use of React Three Fiber, a React renderer for Three.js, addresses the high performance rendering requirements of the project. React Three Fiber enables efficient integration of 3D graphics into the React application while taking advantage of GPU acceleration to render complex scenes such as animated orbital paths, rotating spacecraft, and dynamic uncertainty ellipsoids. This approach ensures real time responsiveness and smooth user interactions even when visualizing large datasets or complex simulations [50].

### *4.3.3. DT Simulator*

The backend DT simulator is built on Python, leveraging the GODOT library for orbital calculations and trajectory predictions [41]. This component is responsible for generating realistic satellite behavior data during development, serving as a substitute for actual satellite telemetry. The simulator processes orbital parameters to generate position and generates random attitude data, power levels, temperatures, and communication link status.

The backend service must also maintain a simulated satellite catalog that mimics the expected structure of a future DT system. While this implementation deliberately omits administrative capabilities for database modification, it provides a RESTful API that supports satellite discovery and selection—crucial functionalities for any satellite monitoring system. This approach allows operators to retrieve a list of available satellites and select specific ones for detailed monitoring, establishing patterns that will seamlessly translate to a real DT implementation. The RESTful API is particularly well-suited for this hierarchical data access pattern, where satellite metadata and historical data can be efficiently queried using standard HTTP methods. For instance, a GET request to '/satellites' provides the catalog listing, while '/satellites/{id}/telemetry' retrieves specific satellite data, following REST conventions that will remain valid when transitioning to a real system.

### *4.3.4. Communication Layer*

The Communication Layer serves as the critical bridge between the frontend visualization components and the backend DT simulator. This layer implements a REST API client that handles all data communication, including configuration data, historical telemetry, and near real-time monitoring through an intelligent polling mechanism. This intelligent polling mechanism is based on the version key system, where data is only transmitted if there is new data to transmit. The client implements features such as request retrying, error handling, and response caching to ensure reliable data access.

The communication protocol of the system is built upon HTTP transport layers, chosen for their widespread support and standardization. This approach leverages the rich ecosystem of tools and caching mechanisms of HTTP [51]. JSON (JavaScript Object Notation) was selected as the message format due to its simplicity, widespread support in both JavaScript and Python, and efficient parsing capabilities [52].

## *4.4. Design Rationale*

The architectural decisions made in this visualization system were driven by specific requirements and careful consideration of alternatives. This section discusses the key design choices, their rationale, and the trade-offs involved.

Several architectural approaches were considered during the design phase of the system. In evaluating GUI development platforms, various options were examined including Unity, a powerful game engine with extensive 3D capabilities and cross-platform support; Unreal Engine, known for advanced real-time 3D rendering and high-quality graphics; Qt, a traditional desktop application framework with OpenGL support; and ultimately React + Three.js, which was selected as the web-based solution with modern development workflow.

For the backend framework, several Python-based options were evaluated including Django, Flask, and FastAPI, as well as Express.js as a JavaScript alternative, ultimately leading to the selection of FastAPI for implementation.

For near real-time communication between the frontend and backend, multiple approaches were explored. These included WebSocket technology, Server-Sent Events (SSE), traditional polling, and GraphQL Subscriptions. After evaluation and consultation with satellite operators, an intelligent polling mechanism was selected as the primary communication approach.

For the message format, Protocol Buffers, MessagePack, JSON, XML and YAML were considered. JSON was selected due to its simplicity and widespread support in JavaScript and Python.

The following criteria guided the architectural decisions:

Development efficiency was a key consideration, focusing on the ability to quickly iterate during development, the availability of robust tooling and debugging support, and the familiarity of the developer with the chosen technologies. Performance requirements encompassed several critical aspects: the system needed to efficiently handle large datasets, provide smooth 3D rendering capabilities, and optimize browser resource utilization. Maintainability was prioritized through emphasis on code organization and modularity, comprehensive testing capabilities, thorough documentation, and provisions for future extensibility. Finally, integration capabilities were essential, requiring seamless integration with future DT systems, support for various data formats used in satellite telemetry.

### *4.4.1. Key Trade-offs*

The selected architecture involves several important trade-offs. These trade-offs were carefully evaluated against the requirements of the project, with particular emphasis on development efficiency and maintainability. The chosen architecture prioritizes developer productivity and system flexibility over absolute performance optimization, recognizing that the primary goal of the visualization system is to provide a robust and extensible platform for satellite monitoring rather than handling extreme-scale data processing.

The decision to use React + Three.js over game engines or desktop frameworks was driven by several compelling advantages. The browser-based deployment eliminates installation requirements and simplifies updates, while modern web technologies provide sufficient 3D rendering capabilities through WebGL. The platform offers an extensive ecosystem of visualization libraries and tools, along with better integration potential for existing web-based satellite control systems. Additionally, React + Three.js presents a simpler learning curve compared to game engines for visualization-focused development, while providing native support for responsive design and cross-platform compatibility [47, 50].

The choice of FastAPI over Django resulted in significant advantages in performance, automatic API documentation generation, and modern asynchronous support. This choice was particularly fitting as GODOT, which powers our backend simulation, is also Python-based. FastAPI ensures seamless integration within the Python ecosystem and simplifies the development and deployment of the backend service. The frontend interacts with these API endpoints to fetch data needed for rendering various visualizations, while the decoupled architecture allows the frontend and backend to evolve independently, supporting modular development and future system expansion. While Django offers a more comprehensive ecosystem with built-in database management and admin interfaces, these features were unnecessary for our simulation-focused backend that simply serves fixed satellite data [53]. The lightweight nature of FastAPI better suited our needs, where the primary requirement was efficient handling of HTTP requests for serving simulated satellite telemetry [54].

For near real-time monitoring, an intelligent polling mechanism was selected over alternative approaches like WebSocket, Server-Sent Events, and GraphQL Subscriptions. This decision was informed by direct consultation with satellite operators, who confirmed that updates every 10 to 30 seconds are more than sufficient for monitoring purposes, particularly given that satellites only communicate during ground station contact windows and asking for telemetry has a cost as the satellite can only communicate so much data during a contact window. While streaming protocols like WebSocket or SSE might offer lower latency, this advantage is unnecessary given the actual operational patterns of satellite communications [55].

The choice of JSON over other message formats was driven by several key advantages in our React-based architecture. Since React is built on JavaScript, JSON data can be directly consumed without additional parsing or transformation steps, reducing processing overhead and simplifying data handling throughout the frontend application. The flexible schema of the format proves particularly valuable during development, allowing for easy addition of new telemetry types or visualization parameters without requiring protocol updates, especially important as visualization requirements evolve with better understanding of operator needs.

Furthermore, the text-based format of JSON significantly facilitates debugging and development by allowing developers to easily inspect message contents in browser developer tools and network monitors. This readability accelerates development and troubleshooting cycles. The format also offers excellent cross-language support—while native to JavaScript, it is well-supported in Python (our backend language) through the standard library and the automatic serialization of FastAPI, ensuring seamless data exchange between frontend and backend components [52].

### *4.5. Development Progress*

The backend is developed in Python, using the FastAPI framework and the GODOT library for orbital calculations and trajectory predictions. It receives API requests with TLEs and returns an array of data in the JSON format containing the position of the satellite in different moments of time and in different reference frames. The reference frame of ICRF is used for 3D visualization while the ITRF is used for 2D ground-track visualization. The frontend is developed in React, and allows visualizing the orbit of multiple satellites as they are added to the visualization system. This is done by asking the user to provide a TLE, which is then sent to the backend and leads to the generation of a trajectory to be presented in the ground-track mode. The visualization of the orbit in 3D was achieved when the software supported a single satellite, and is yet to be extended to multiple satellites. Complementary to the orbit visualization, there is also a timeline, for selecting the moment of time to be visualized and an extendable list of satellites.

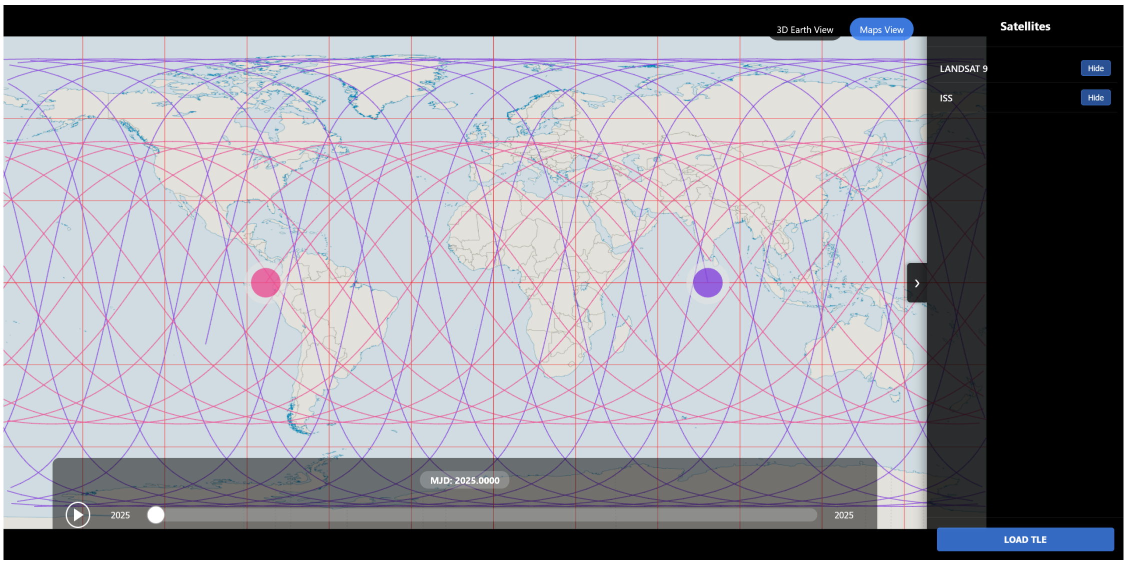

Figure 4: Visualization of the ground-track of two distinct orbital paths

As can be seen in Figs. 4 and 5, the visualization of the ground-track of two distinct orbital paths is achieved by adding two different TLEs to the visualization system.

## 5. Work Plan

In this section, the work plan is presented, which entails the development of the frontend and backend, as well as the communication layer and the planning of future interviews.

### *5.1. Frontend*

Before moving on to developing additional visualization modes, such as the attitude visualization, the timeline must be fixed to adapt its limits, by which is meant the beginning time and end time of the slider, to the interval of the most recently added satellite. It would

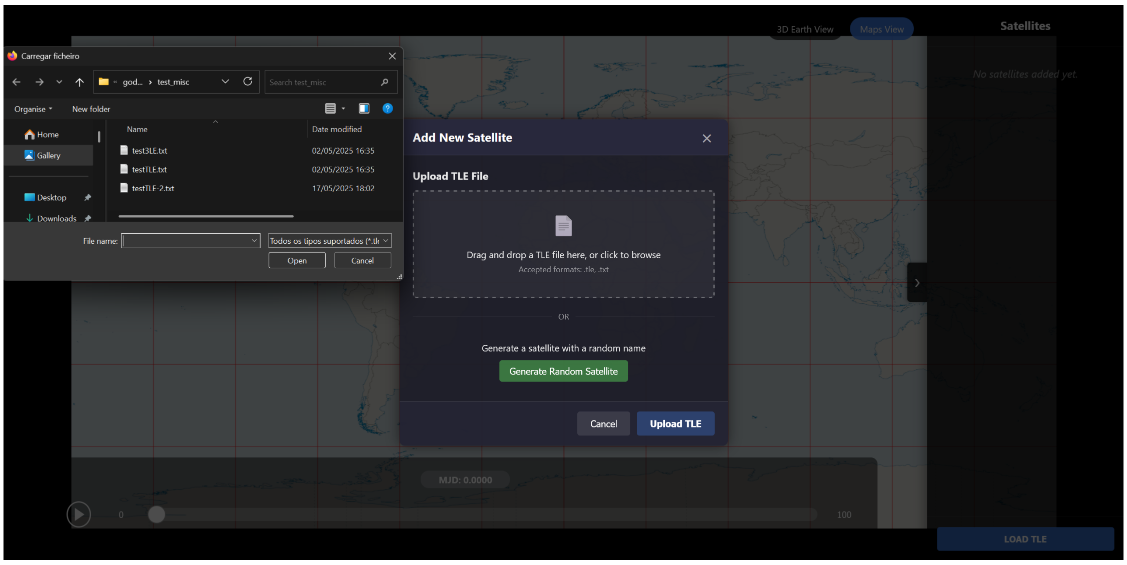

Figure 5: Window to load a TLE file

also be crucial to provide the option to the user, when adding a satellite, of selecting the time interval of the data to be requested from the backend. The list of satellites must not only provide the option to delete a satellite, but also to select a satellite to be visualized in 3D attitude mode. Finally, the information about the added satellites must be stored in cache to avoid having to request the same data multiple times or losing it when the page is refreshed.

After fixing the ground-track visualization and adding the necessary features, what is to be developed first in the frontend is the attitude visualization, which will be interlinked with the 3D orbit visualization as the Earth and the orbit should be visualized in 3D in the same view as the vectors that represent the attitude of the satellite. Representing any given vector in 3D is trivial using React Three Fiber. One challenge however is being able to represent the Earth in the correct state of rotation given any time and date. For this GODOT will most likely be used to provide the direction of the axis of rotation of the earth and the hour of the day will be used to rotate the Earth around its axis.

Additionally, and still on the topic of attitude visualization, the celestial map is yet to be developed. Some research must still be done on the advantages and disadvantages of different types of projection maps, although the equirectangular projection seems to be the easiest to implement and the one which can best fit in a visual interface which is a rectangular screen.

The integration of a dashboard is also yet to be attempted. If the attempt to integrate Grafana fails, the functionalities of dashboard and time series visualization will be developed, which is possible to be done using React. While this second option might provide fewer features than Grafana it will necessarily have to achieve the ability to select points in the time series and to visualize the position and attitude of the satellite at the selected time.

Of yet lesser priority is not only the visualization of the packages transmitted by each satellite, but also the visualization of the environmental data, and more specific visualization features such as the field of view of a sensor or satellite projected onto the Earth or the ability to choose the color of a given trajectory.

### *5.2. Backend, Communication Layer and Future Interviews*

The development of the backend will accompany the development of the frontend. As a new feature is developed in the frontend which requires data from the backend, the backend will be developed to provide such data. This will include a directory structure to represent the data available in the backend, which means a list of satellites, each containing TLEs, the possibility to obtain trajectory data, attitude data, and more.

Concurrently, the communication layer has already been developed, and is based on the use of the FastAPI framework. However, the goal is to develop an interface in the frontend such that other communication protocols can be used, such as WebSockets, and to abstract the communication layer from the rest of the application. This will likely be done early in the development in order to avoid having to refactor the entire application to use a different communication protocol.

The discussion with the ISTSAT team will be maintained as the project progresses. Additionally, more satellite operators will be interviewed, with the possibility of allowing them to test an early version of the visualization system and give meaningful feedback.

## 6. Conclusion

The development of effective visualization tools for satellite DTs responds to a clear need identified in both academic literature and operational practice. While DTs have advanced significantly in modeling, simulation, and diagnostics, their ability to convey complex information in an intuitive and actionable way remains underexplored. This project addresses that gap by focusing not on the internal mechanics of the DT, but on the interface that mediates human understanding of satellite state, behavior, and context. The system is shaped by feedback from satellite operators and existing visualization tools, ensuring that the solution is grounded in real-world needs rather than theoretical assumptions. By adopting a modular, web-based architecture, the work also promotes flexibility, scalability, and accessibility, allowing future integration with a variety of DT backends and mission scenarios. The result is a framework that not only fills a missing layer in current DT implementations but also serves as a foundation for future research and development in satellite visualization. The following subsection outlines the concrete achievements and contributions made during the course of this work.

### *6.1. Achievements and Contributions*

The achievements of this project include the planning of the development of the visualization system, which required a thorough analysis of the requirements and constraints of the project, but also include the development of a frontend application which achieves some visualization features and a backend which simulates some of the behavior of a DT, as well as the communication layer. Both the discussion with the ISTSAT team and the interviews with the satellite operators have been crucial to the success of the project, as they provided valuable feedback and insights that were used to redirect the development of the software.

When the software is completed by the end of the thesis, the contributions of the author will be the development of an open source visualization system for satellite operators, which will be able to communicate with DTs for updates on the state of the satellite.

## References

[1] D. Shangguan, L. Chen, J. Ding, A Digital Twin-Based Approach for the Fault Diagnosis and Health Monitoring of a Complex Satellite System, Symmetry 12 (8) (2020) 1307. doi:10.3390/sym12081307.

[2] I. M. Ferreira, P. J. Gil, Integration of different visualisations to hide the complexity on the design of space systems: 62nd International Astronautical Congress 2011, IAC 2011, 62nd International Astronautical Congress 2011, IAC 2011 (2011) 7246–7260.

[3] AI Fights Space Debris - Neuraspace, https://www.neuraspace.com/company, [Accessed: 26-Jun-2025].

[4] Q. Li, Overview of Data Visualization, Embodying Data (2020) 17–47doi:10.1007/978-981-15-5069-0_2.

[5] M. Friendly, H. Wainer, A History of Data Visualization and Graphic Communication, Harvard University Press, 2021.

[6] A. Lavanya, S. Sindhuja, L. Gaurav, W. Ali, A Comprehensive Review of Data Visualization Tools: Features, Strengths, and Weaknesses, International Journal of Computer Engineering in Research Trends 10 (1) (2023) 10–20. doi:10.22362/ijcert/2023/v10/i01/v10i0102.

[7] R. Rosen, G. Von Wichert, G. Lo, K. D. Bettenhausen, About The Importance of Autonomy and Digital Twins for the Future of Manufacturing, IFAC-PapersOnLine 48 (3) (2015) 567–572. doi:10.1016/j.ifacol.2015.06.141.

[8] E. Glaessgen, D. Stargel, The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles, in: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference&lt;BR&gt;20th AIAA/ASME/AHS Adaptive Structures Conference&lt;BR&gt;14th AIAA, American Institute of Aeronautics and Astronautics, Honolulu, Hawaii, 2012. doi:10.2514/6.2012-1818.

[9] P. Jain, J. Poon, J. P. Singh, C. Spanos, S. R. Sanders, S. K. Panda, A Digital Twin Approach for Fault Diagnosis in Distributed Photovoltaic Systems, IEEE Transactions on Power Electronics 35 (1) (2020) 940–956. doi:10.1109/TPEL.2019.2911594.

[10] F. Tao, M. Zhang, Y. Liu, A. Y. C. Nee, Digital twin driven prognostics and health management for complex equipment, CIRP Annals 67 (1) (2018) 169–172. doi:10.1016/j.cirp.2018.04.055.

[11] J. Wang, L. Ye, R. X. Gao, C. Li, L. Zhang, Digital Twin for rotating machinery fault diagnosis in smart manufacturing, International Journal of Production Research 57 (12) (2019) 3920–3934. doi:10.1080/00207543.2018.1552032.

[12] W. Yang, Y. Zheng, S. Li, Application Status and Prospect of Digital Twin for On-Orbit Spacecraft, IEEE Access 9 (2021) 106489–106500. doi:10.1109/ACCESS.2021.3100683.

[13] F. Cravidão, J. P. Figueiredo, J. P. Monteiro, P. J. S. Gil, R. Ventura, Building Data-Driven Satellite Digital Twins, in: IAF Space Systems Symposium, International Astronautical Federation (IAF), Milan, Italy, 2024, pp. 1047–1061. doi:10.52202/078372-0105.

[14] D. L. Iverson, R. Martin, M. Schwabacher, L. Spirkovska, W. Taylor, R. Mackey, J. P. Castle, V. Baskaran, General Purpose Data-Driven Monitoring for Space Operations, Journal of Aerospace Computing, Information, and Communication 9 (2) (2012) 26–44. doi:10.2514/1.54964.

[15] P. Korenhof, E. Giesbers, J. Sanderse, Contextualizing realism: An analysis of acts of seeing and recording in Digital Twin datafication, Big Data & Society 10 (1) (2023) 20539517231155061. doi:10.1177/20539517231155061.

[16] M. Wahal, M. Mahmoudi, A. Bahssain, I. Rekabi, A. Saeed, M. Alzarif, M. Ellethy, N. Elsayed, M. Abdelsalam, T. ElBatt, A Proposed Immersive Digital Twin Architecture for Automated Guided Vehicles Integrating Virtual Reality and Gesture Control:, in: Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, SCITEPRESS - Science and Technology Publications, Porto, Portugal, 2025, pp. 267–275. doi:10.5220/0013176900003912.

[17] W. Hu, T. Zhang, X. Deng, Z. Liu, J. Tan, Digital twin: A state-of-the-art review of its enabling technologies, applications and challenges, Journal of Intelligent Manufacturing and Special Equipment 2 (1) (2021) 1–34. doi:10.1108/JIMSE-12-2020-010.

[18] A. Haghshenas, A. Hasan, O. Osen, E. T. Mikalsen, Predictive digital twin for offshore wind farms, Energy Informatics 6 (1) (2023) 1. doi:10.1186/s42162-023-00257-4.

[19] T. Sun, X. He, Z. Li, Digital twin in healthcare: Recent updates and challenges, DIGITAL HEALTH 9 (2023) 20552076221149651. doi:10.1177/20552076221149651.

[20] M. Jafari, A. Kavousi-Fard, T. Chen, M. Karimi, A Review on Digital Twin Technology in Smart Grid, Transportation System and Smart City: Challenges and Future, IEEE Access 11 (2023) 17471–17484. doi:10.1109/ACCESS.2023.3241588.

[21] D.-G. J. Opoku, S. Perera, R. Osei-Kyei, M. Rashidi, Digital twin application in the construction industry: A literature review, Journal of Building Engineering 40 (2021) 102726. doi:10.1016/j.jobe.2021.102726.

[22] R. Stevens, Digital Twin for Spacecraft Concepts, in: 2023 IEEE Aerospace Conference, 2023, pp. 1–7. doi:10.1109/AERO55745.2023.10115665.

[23] Y. Ye, Q. Yang, F. Yang, Y. Huo, S. Meng, Digital twin for the structural health management of reusable spacecraft: A case study, Engineering Fracture Mechanics 234 (2020) 107076. doi:10.1016/j.engfracmech.2020.107076.

[24] W. Yang, Z. , Yu, S. and Li, Digital twin of spacecraft assembly cell and case study, International Journal of Computer Integrated Manufacturing 35 (3) (2022) 263–281. doi:10.1080/0951192X.2021.1992657.

[25] H. Lei, Z. Fanli, W. Wei, S. Shuai, Z. Haocheng, Digital twin method and application practice of spacecraft system driven by mechanism data, Digital Twin 4 (2024) 2. doi:10.12688/digitaltwin.17913.1.

[26] W. Liu, M. Wu, G. Wan, M. Xu, Digital Twin of Space Environment: Development, Challenges, Applications, and Future Outlook, Remote Sensing 16 (16) (2024) 3023. doi:10.3390/rs16163023.

[27] Y. Peng, X. Zhang, Y. Song, D. Liu, A Low Cost Flexible Digital Twin Platform for Spacecraft Lithium-ion Battery Pack Degradation Assessment, in: 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2019, pp. 1–6. doi:10.1109/I2MTC.2019.8827160.

[28] N. Christofi, X. Pucel, A novel methodology to construct digital twin models for spacecraft operations using fault and behaviour trees, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 473–480. doi:10.1145/3550356.3561550.

[29] Z. Lyu, J. Guo, R. Lou, H. Lv, Artificial Intelligence Based Spacecraft Resilience Optimization in Space Informatics Digital Twins, IEEE Transactions on Aerospace and Electronic Systems 61 (2) (2025) 1834–1847. doi:10.1109/TAES.2024.3459879.

[30] General Mission Analysis Tool (GMAT) v.R2016a(GSC-17177-1) | NASA Software Catalog, https://software.nasa.gov/software/GSC-17177-1, [Accessed: 26-Apr-2025].

[31] Analyzing Spacecraft Attitude Profiles with Satellite Scenario, https://www.mathworks.com/help/aeroblks/analyzing-spacecraft-attitude-profiles-with-satellite-scenario.html, [Accessed: 26-Apr-2025].

[32] Timeloop, https://timeloop.fr/vts/, [Accessed: 23-Jun-2025].

[33] Satellite Tracking System: Orbitron by Sebastian Stoff / Satellite tracking easiest ever!, https://www.stoff.pl/, [Accessed: 26-Apr-2025].

[34] Gpredict: Free, Real-Time Satellite Tracking and Orbit Prediction Software, https://oz9aec.dk/gpredict/, [Accessed: 08-May-2025].

[35] Grafana: The open and composable observability platform, https://grafana.com/, [Accessed: 03-Jun-2025].

[36] Vires for swarm, https://vires.services/, [Accessed: 23-Jun-2025].

[37] https://www.ansys.com/content/dam/amp/2022/june/webpage-requests/stk-product-page/brochures/stk-overview-brochure.pdf, https://www.ansys.com/content/dam/amp/2022/june/webpage-requests/stk-product-page/brochures/stk-overview-brochure.pdf, [Accessed: 15-Mar-2025].

[38] Advanced Space Mission Visualization (TRACK) | satsearch, https://satsearch.co/products/terma-advanced-mission-visualization-track, [Accessed: 16-Mar-2025].

[39] V. Tan, J. L. Labrador, M. Caesar Talampas, MATA: Mission, Attitude, and Telemetry Analysis

Software for Micro-Satellites, in: 2020 IEEE REGION 10 CONFERENCE (TENCON), 2020, pp. 614–619. doi:10.1109/TENCON50793.2020.9293937.

[40] 42: A Comprehensive General-Purpose Simulation of Attitude and Trajectory Dynamics and Control of Multiple Spacecraft Composed of Multiple Rigid or Flexible Bodies(GSC-16720-1) | NASA Software Catalog, https://software.nasa.gov/software/GSC-16720-1, [Accessed: 16-Mar-2025].

[41] Welcome to GODOT! — GODOT 1.11.0 documentation, https://godot.io.esa.int/godotpy/, [Accessed: 08-May-2025].

[42] J. P. Monteiro, A. Cunha, A. Silva, C. Fernandes, D. Neves, F. Naf, G. Tavares, J. Freitas, J. Pinto, M. Piedade, N. Ramos, P. J. S. Gil, P. Macedo, R. Afonso, R. Encarnação, R. Ramos, T. Almeida, R. M. Rocha, ISTSat-1, a space-based Automatic Dependent Surveillance-Broadcast demonstration CubeSat mission, International Journal of Satellite Communications and Networking 40 (4) (2022) 268–293. doi:10.1002/sat.1440.

[43] PlotJuggler, https://plotjuggler.io, [Accessed: 03-Jun-2025].

[44] SatDump, https://www.satdump.org/, [Accessed: 03-Jun-2025].

[45] Gqrx SDR – Open source software defined radio by Alexandru Csete OZ9AEC, [Accessed: 03-Jun-2025].

[46] E. J. Daly, G. Drolshagen, A. Hilgers, H. D. R. Evans, Space Environment Analysis: Experience and Trends, in: Environment Modeling for Space-Based Applications, Vol. 392, 1996, p. 15.

[47] C. Gackenheimer, Introduction to React, Apress, 2015.

[48] M. Biehl, RESTful API Design, API-University Press, 2016.

[49] P. Bourque, R. Fairley, Guide to the Software Engineering Body of Knowledge - SWEBOK V3.0, 2014.

[50] Introduction - React Three Fiber, https://r3f.docs.pmnd.rs/getting-started/introduction, [Accessed: 09-May-2025].

[51] L. Popa, P. Wendell, A. Ghodsi, I. Stoica, HTTP: An Evolvable Narrow Waist for the Future Internet.

[52] T. Bray, The JavaScript Object Notation (JSON) Data Interchange Format, Request for Comments RFC 8259, Internet Engineering Task Force (Dec. 2017). doi:10.17487/RFC8259.

[53] Django documentation | Django documentation, https://docs.djangoproject.com/en/5.2/, [Accessed: 26-May-2025].

[54] FastAPI, https://fastapi.tiangolo.com/, [Accessed: 26-May-2025].

[55] R. Appelqvist, O. Örnmyr, Performance comparison of XHR polling, Long polling, Server sent events and Websockets.