ATILIM ÜNİVERSİTESİ

2024-25 Fall

SE 494 Final Project Report

RTSG

Project Short Title	RTSG		
Project Full Title	Radar and Track Scenario Generator		
Subject Classification	Engineering/Scientific Software		
Student name(s) (Team members of project)	Samet Çolak		
Full Name of Corporation	AYDIN YAZILIM ve ELEKTRONİK SANAYİ A.Ş.		
Mentor Name(s) (Corporation member)	MURAT AYKAN		
Coordinator Name(s) (University member)	ALİ YAZICI		
Adı Soyadı Tarih İmza	Öğrenci	Firma Ortak Eğitim Danışmanı	Atılım Üniversitesi Ortak Eğitim Koordinatörü

ATILIM ÜNİVERSİTESİ

Table of Contents

Abstract

The Radar Track Scenario Generator (RTSG) project addresses the challenge of generating and managing radar and moving object data for testing track fusion algorithms. By utilizing RTSG, realistic scenarios are created by integrating radar and flight data to simulate real-world conditions. The tool enables the creation of various scenarios in which radar and moving object tracks are generated, providing valuable datasets for testing the accuracy and performance of track fusion algorithms. This process is essential for evaluating algorithms in fields like air traffic control, military operations, and civilian tracking applications.

The RTSG approach involves generating synthetic data based on predefined scenarios, where radar and flight information are incorporated to mimic real-life environments. The generated data is then used to validate track fusion algorithms, ensuring their correct functionality. This unique approach allows for testing track fusion systems under controlled but realistic conditions, making the results reliable for real-world applications. By facilitating scenario creation and testing, the RTSG project provides a significant contribution to the development of robust and accurate tracking systems.

The originality of this work lies in its ability to simulate a wide range of radar and flight scenarios, which are crucial for testing complex track fusion algorithms. The use of realistic data ensures that these algorithms are thoroughly evaluated, improving their overall performance. The benefits of this project extend to society by enhancing the efficiency and accuracy of radar systems used in air traffic management, military surveillance, and civilian applications, leading to safer and more reliable systems.

Keywords

Data Generation, Radar, Flight, Track Fusion, Scenario Generator

Acknowledgements

I would like to express my deepest appreciation to all those who made it possible for me to complete this project. A special thanks goes to my senior project supervisor, Mr. Ali Yazıcı, the Internship Coordinator at Atılım University, for his invaluable guidance and support throughout the project. His thorough review of reports and direction helped me create a more accurate and well-structured report. I am also deeply grateful to my project mentor, Mr. Murat Aykan, the Internship Advisor at the company, for his continuous encouragement and for taking the time to hold necessary meetings and provide valuable support throughout the project. My sincere appreciation goes to my development team leader, Mr. Fatih Aydın, for his technical expertise and leadership, which were crucial for overcoming development challenges. His guidance was invaluable whenever I encountered difficulties in the development process. Lastly, I would like to thank Mr. Mert Sungur, the resource manager, for ensuring that the project had all the necessary tools and support for success. His effective management of resources played a vital role in the smooth progression of the project. Each of these individuals has contributed significantly to the completion of this project, and I am truly grateful for their support.

Acronyms

RTSG	Radar and Track Scenario Generator
IDE	Integrated Development Environment
GUIs	Graphical User Interfaces
CI	Continuous Integration
CD	Continuous Deployment
JDK	Java Development Kit
WP	Work Package
TDD	Test-Driven Development
UCD	User-Centered Design
IP	Intellectual Property

Ch1: Introduction

1.1 Problem Statement

Creating and managing radar and moving object tracks can be a challenging process, especially when testing fused track algorithms. The RTSG (Radar and Track Scenario Generator) project aims to solve this problem. This project enables the user to create various scenarios and track radar and moving object tracks in these scenarios. Creating such scenarios accurately and efficiently is important for test engineers to evaluate the performance of their algorithms. In daily life, such tests create significant impacts in areas such as air traffic control, military exercises and civilian applications.

1.2 Scope

The RTSG project aims to create and manage different scenarios using radar and moving object tracks. The project focuses on enabling test engineers to create scenarios that test fused track algorithms. It facilitates the identification of radar and moving object tracks and the development of scenarios where these tracks can be followed. Additionally, the project supports saving, editing, and reusing the created scenarios while ensuring efficient management of radar and moving object tracks through a user-friendly interface.

Ch2: Project Plan

2.1 Methodology

The RTSG (Radar and Track Scenario Generator) project will be developed in two increments using Incremental Methodology. The main purpose of the incremental methodology is to divide the project into small parts, add to the product at each stage, and increase the accuracy of the system with the feedback obtained. This is a great advantage, especially in projects that are complex and sensitive to user feedback. Each increment in the RTSG project is designed to improve the functionality and user experience of the system. The improvements made in the two increments and the purpose of each are as follows:

Initial Increment (First Release):

The first increment focuses on the core functionality of the RTSG application, aiming to create a prototype where users can identify radar and moving object tracks, create scenarios, and save them. This initial release includes the identification of radar and moving tracks, the creation of scenarios, and the management of radar tracks with basic user interface functionalities. After testing the basic functionality, user feedback will be collected, which will guide the development of the second release. Based on this feedback, new features will be introduced to enhance the accuracy of existing features and improve the overall user experience.

Second Increment (Final Release):

The second increment is developed based on the feedback from the first increment, completing the system's functionality and enhancing the user experience. Furthermore, the application undergoes a broader testing process. This increment includes improved and optimized radar and track scenario management, enhanced ease of use and performance with a more user-friendly interface, and more comprehensive testing and performance optimizations. Additionally, new features are integrated based on user feedback, such as improved scenario management and greater user control.

Purpose of Incremental Methodology:

The feedback collected at the end of the first increment will guide the improvements to be made in the second increment, making the final version of the system more accurate. The incremental methodology increases the quality of the project with each improvement, producing a product that better meets the needs of end users. Additionally, the quick implementation of feedback from the first release allows for major improvements in the final product. These two augmentation processes were designed to ensure that the RTSG project's final product is more accurate and efficient. In both increments, improvements will be made based on development, testing, and user feedback, ensuring the best outcome for the project.

2.2. Time Schedule (Gantt Chart)

WP / Task No	WP / Task Name	Start Date	End Date	Weeks																				
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
WP0 / T0	Project Management	20/08/2024	10/01/2025	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
WP1 / T1	Requirement Analysis	20/08/2024	26/08/2024	■																				
WP1 / T2	Design	26/08/2024	09/09/2024		■	■																		
WP1 / T3	Development	09/09/2024	18/11/2024			■	■	■	■	■	■	■	■	■	■	■								
WP1 / T4	Testing	11/11/2024	25/11/2024												■	■	■							
WP1 / T5	Feedback	25/11/2024	02/12/2024														■	■						
WP2 / T1	Requirement Update	02/12/2024	09/12/2024															■	■					
WP2 / T2	Design	09/12/2024	16/12/2024																■	■				
WP2 / T3	Development	16/12/2024	06/01/2025																	■	■	■	■	
WP2 / T4	Testing	06/01/2025	09/01/2025																				■	■
WP2 / T5	Feedback	09/01/2025	10/01/2025																					■

2.3. Deliverables List

ID	**Type**	**Description**	**Due Date**
D1	Document	Project Plan: Comprehensive plan outlining project scope, timeline, and resources.	20/08/2024
D2	Document	Requirements Specification: Detailed specifications of system requirements.	27/08/2024
D3	Document	Design Document: Architectural design and design patterns used.	10/09/2024
D4	Software	Prototype of Radar Detection System: Initial version of the radar detection functionality.	18/11/2024
D5	Document	Test Plan: Plan detailing the testing strategy and scope.	02/12/2024
D6	Software	First Build of Application: First complete build of the app for review.	02/12/2024
D7	Document	Feedback Report: Report summarizing feedback received from the first build.	09/12/2024
D8	Document	Updated Requirements Document: Revised requirements based on feedback.	16/12/2024
D9	Software	Final Build of Application: Final version of the app, incorporating all updates and fixes.	06/01/2025
D10	Document	User Manual: Comprehensive guide for users on how to operate the application.	07/01/2025
D11	Document	Final Project Report: Detailed report summarizing project outcomes, lessons learned, and future recommendations.	10/01/2025

Ch3: Literature Review

In this chapter, we will explore the existing literature relevant to the RTSG (Radar and Track Scenario Generator) project. We will identify the foundational references that have informed our project and discuss how our work relates to and builds upon them. The literature review is organized into the following subsections:

3.1 Radar Systems and Technology

Radar systems have been a critical component in various fields such as aviation, maritime, and military applications. The study of radar technology provides a foundational understanding necessary for developing radar-based applications like RTSG [1][2]. These two studies offer in-depth knowledge on radar principles and technologies that underpin our project’s capability to accurately simulate radar and track scenarios.

3.2 Track and Data Fusion Algorithms

Track and data fusion algorithms are essential for combining data from multiple radar sources to form a coherent picture of object movements. Our project uses these algorithms to test fused track scenarios [3][4]. These sources provide fundamental theories and practices for data fusion, directly informing the algorithms tested by our RTSG application.

3.3 Scenario Generation and Simulation

The ability to generate and simulate scenarios is critical for testing and evaluation purposes. Our project draws on existing techniques for scenario generation to create realistic and testable environments [5][6]. These texts offer insights into simulation modeling that are directly applicable to our project’s scenario generation functions.

3.4 Use Cases in Radar and Track Scenario Applications

Examining existing applications of radar and track scenarios helps contextualize our project within the broader field and highlights its practical relevance [7][8]. These references explore practical applications of radar systems and provide a basis for understanding how RTSG can be used in real-world scenarios.

Ch4: Increment 1

4.1 Requirements

4.1.1 Overall Description of the Project

4.1.1.1 Product Perspective

RTSG (Radar and Track Scenario Generator) will be a standalone software that creates various scenarios using radar and moving object tracks. This product allows test engineers to easily create scenarios in which they can evaluate the performance of fused track algorithms. RTSG can operate independently, without directly integrating with any external system or application. The data generated by the product can be integrated into a larger test system, but itself is independent and modular.

4.1.1.2 Product Functions

RTSG is designed to handle several core functions essential for creating and managing radar and moving object scenarios. It allows users to define radar and moving object tracks for the creation of new scenarios. Users can open, edit, and delete existing scenarios through the scenario management function. The radar management feature enables the addition, modification, and removal of radar information, including location, scanning radius, and error factor. The flight management function provides the capability to manage moving object tracks, allowing users to add, edit, or delete information about the position, speed, direction, and other relevant parameters. Event management allows the addition and modification of events that involve changes to the speed and direction of moving objects. Finally, RTSG generates data output based on the defined scenarios, supporting test and analysis needs.

4.1.1.3 User Characteristics

The RTSG is designed with the objective of supporting test engineers and professionals in radar monitoring and analysis, who typically have extensive technical backgrounds. The target users hold a bachelor's or master's degree in engineering or technical disciplines. They possess experience in radar systems, air traffic control, or related fields, and are skilled in using various technical systems. Additionally, users have expertise in software development, particularly in languages such as Java and JavaFX, and are proficient in data analysis and processing.

4.1.1.4 Constraints

The RTSG project will be subject to several limitations. First, it must comply with regulatory policies, particularly concerning data security and resilience. The application will also need to operate within specified hardware limitations, ensuring compatibility with certain hardware configurations. Additionally, any updates or changes in the software development process must adhere to standard compliance requirements. Furthermore, the integration with other systems or applications must be evaluated and defined based on specific requirements.

4.1.1.5 Assumptions and Dependencies

The RTSG project will be developed with the assumption that the necessary hardware and operating system are already determined and available for deployment. Additionally, it is expected that the data formats used for radar and moving object tracks will be standardized and widely accepted by all users. Furthermore, it is assumed that the target users will have the required training and experience to effectively use the software, ensuring smooth operation and functionality.

4.1.2 Specific Requirements

4.1.2.1 External Interface Requirements

This section will provide a detailed description of all inputs into and out of the RTSG project.

1. Item Name

• Radar Information Entry

• Moving Object (Flight) Information Entry

• Event Information Entry

• Scenario Files Entry

• Data Output Files

2. Definition of Purpose

• Radar Information Entry: Entering information such as location, scanning radius, starting angle and error factor of the radars.

• Moving Object (Flight) Information Entry: Entering information such as location, speed, direction and height of moving objects.

• Event Information Entry: Entering events indicating the speed and direction changes of moving objects.

• Scenario Files Entry: Uploading scenario files created or edited by the user to the system.

• Data Output Files: Receiving data files produced as a result of running the scenarios.

3. Source of Input or Target of Output

• Input Source: User input, existing scenario files

• Target of Output: User, data analysis software, other systems

4. Valid Range, Accuracy and/or Tolerance

• Radar Information: Lat/Lon: ±0.0001 degrees, Scanning Radius: ±5 meters

• Moving Object Information: Lat/Lon: ±0.0001 degrees, Speed: ±1 m/s

• Event Information: Time: ±0.1 seconds

5. Units of Measurement

• Position: Degrees (decimal degrees)

• Speed: Meters per second (m/s)

• Height: Meters (m)

• Time: Seconds (s)

6. Timing

• All inputs are received during scenario creation or editing and data outputs are produced when the scenario is run.

7. Relationships with Other Inputs/Outputs

• Radar and moving object information are used together as part of the scenario.

• Event information depends on moving object information.

8. Screen Formats/Organization

• User-friendly interface, form-based login screens

9. Window Formats/Organization

• Pop-up windows, data entry windows, Visual map window

10. Data Formats

• Radar, moving object and event information is stored in TXT.

• Data outputs are stored as TXT files.

11. Command Formats

• Commands are issued through clicks and selections from the user interface.

12. Ending Messages

• Messages informing the user of the successful/current status are displayed.

4.1.2.2 Functional requirements

Functional requirements define the basic actions that must be performed for the RTSG to accept and process inputs and produce output.

a) Validity Checks of Entries

• The system checks the validity of the entered radar and moving object information (for example, whether the lat/lon values are within the appropriate range).

b) Complete Sequence of Operations

1. User enters and saves radar information.

2. The user enters and saves moving object information.

3. The user enters and saves event information.

4. The user runs the scenario.

5. The system generates data based on radar and moving object information.

c) Reactions to Abnormal Situations

1. Overflow: Shows an error message if the input data exceeds its limits.

2. Communication Facilities: Warns in case of data input/output errors of the system.

3. Error Management and Recovery: Provides warnings to the user for incorrect entries and the opportunity to correct them.

d) Effect of Parameters

• Changes in input parameters affect the outcome of the scenario.

e) Relationship of Outputs to Inputs

1. Input/Output Sequences: Sequential outputs are produced depending on the input data.

2. Input to Output Conversion Formulas: Mathematical operations based on moving object and radar information.

4.1.2.3 Quality Requirement (HW, OS, Database, programming environment, performance, security, reliability, etc)

All quality requirements established for RTSG are specified in this section.

a) Special Hardware Requirement

• Standard computer hardware (CPU, RAM) will be sufficient.

b) Special Operating System Requirement

• It can run on Windows operating systems.

c) Database Requirements

• There was no need to use a database.

d) Special Programming Environment Requirement

• Java programming language and JavaFX library.

e) Performance Condition

• Real-time data processing and fast data input/output should be provided.

f) Security Requirements

• RTSG Project is not need security requirements.

g) Availability Requirement

• User-friendly interface and easy navigation should be provided.

4.2 Design

4.2.1 Software Architecture

4.2.1.1 Actors

The actors defined in the RTSG (Radar and Track Scenario Generator) project are:

User (Test Engineer): Creates, edits and runs scenarios. Enters, updates and deletes radar and moving object information.

System: Performs data processing, scenario management and data output generation based on user input. Stores and manages scenario, radar, flight and event information.

4.2.1.2 Constraints

Constraints shaping the architecture and design of the RTSG project:

Hardware Restrictions: The software must run on standard computer hardware (for example, a certain CPU and RAM capacity).

Operating System Restrictions: The software must be run on Windows operating systems.

Performance Constraints: Real-time data processing and fast data input/output must be provided.

Standard Compliance: Software development standards and coding guidelines must be followed.

4.2.1.3 Software Architecture

The software architecture of the RTSG project consists of user interface, business logic and database components. These components interact with each other through interfaces.

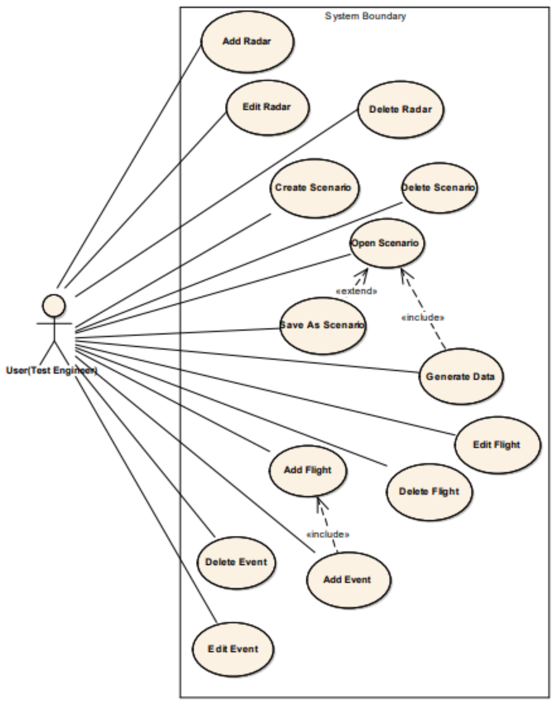

Figure 1 : Use Case Diagram

The use case diagram shown in Figure 1 illustrates the actions that test engineers can perform within the RTSG application. The diagram highlights key system functions, including adding flights, adding radars, and generating data. These functions are essential for the test engineers to create and manage test scenarios, evaluate radar and flight tracks, and test the track fusion algorithms effectively.

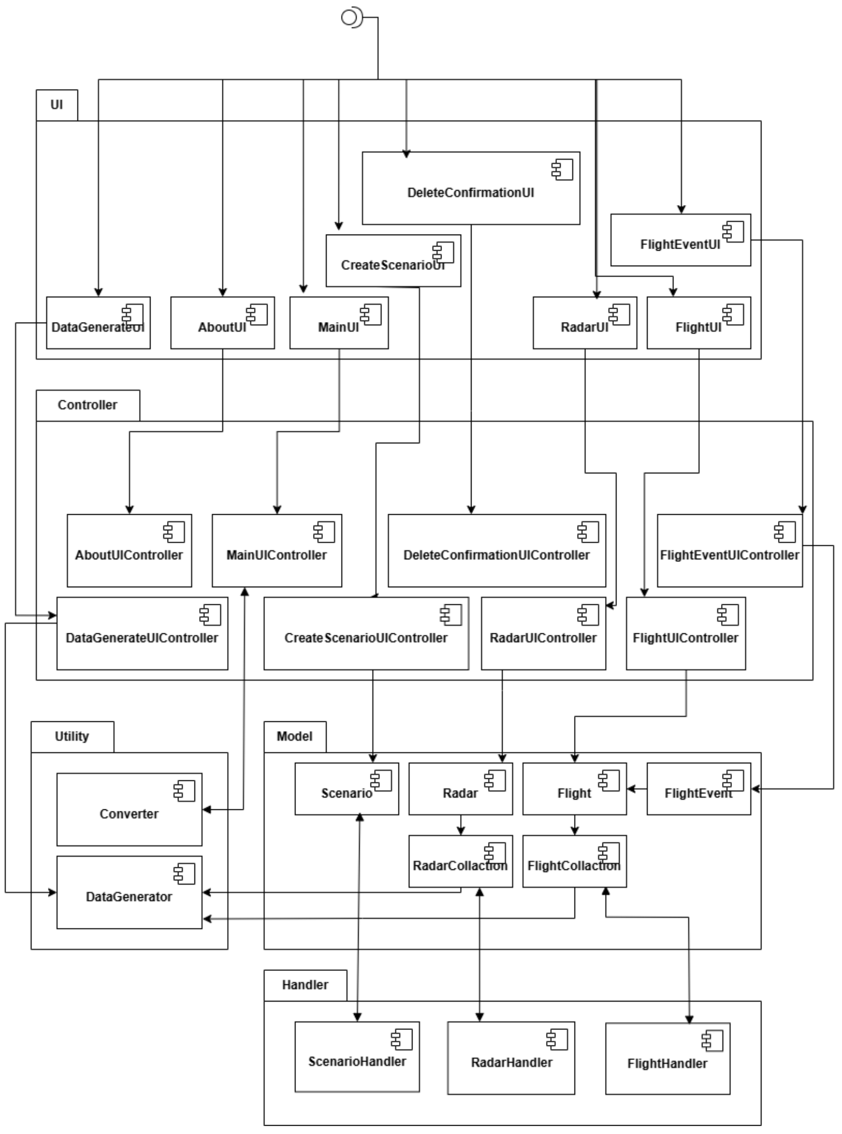

Figure 2 : Component Diagram

The component diagram shown in Figure 2 illustrates how the different components of the RTSG project are organized and their relationships. This diagram provides a clear view of how various modules, such as the radar and flight management systems, data generation components, interact with each other within the application. It highlights the system's modular structure, ensuring that each component functions independently while contributing to the overall system's performance and efficiency. The component diagram is a crucial part of understanding the architecture of the project and how each element supports the functionality of the RTSG application.

4.2.2 Views

4.2.2.1 Logical View

a) Class diagram

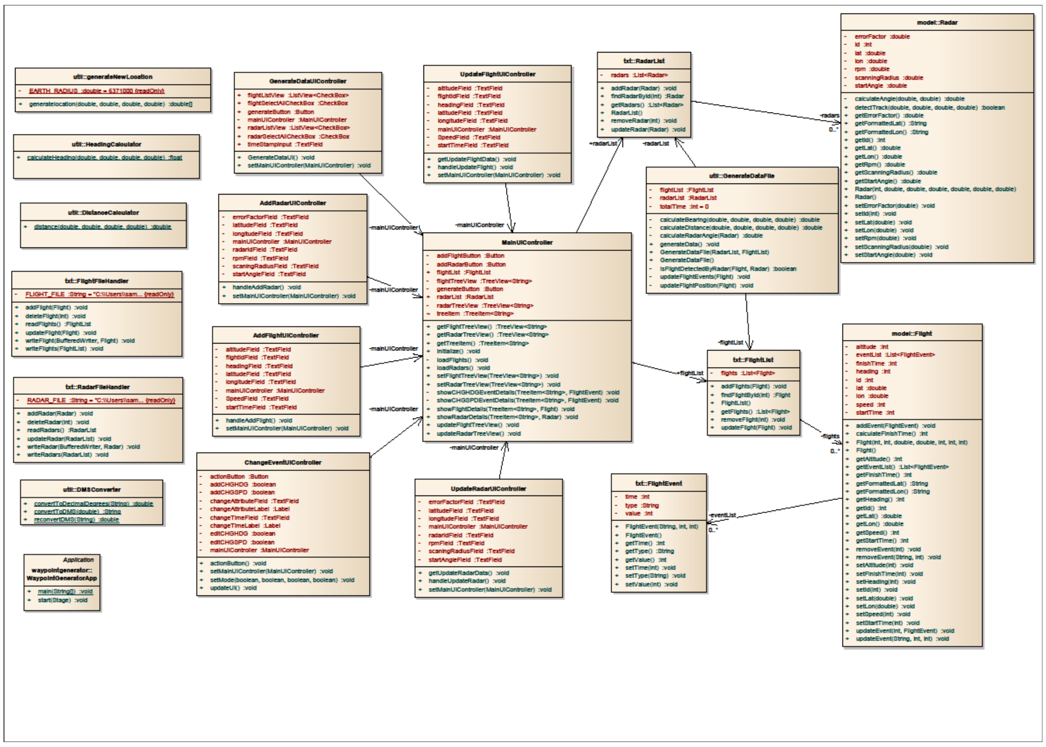

Figure 3 : Class Diagram

The class diagram shown in ***Figure 3*** demonstrates the relationships between the classes in the RTS**G** project and their attributes. This diagram provides a detailed view of how the various classes are structured within the system, including their attributes and methods. It illustrates how classes are interconnected, allowing for efficient data flow and interaction within the application. The class diagram is essential for understanding the object-oriented design of the project, providing a clear representation of the system's structure and helping developers ensure the correct implementation of the system's components.

4.2.2.2 Process View

a) Sequence diagrams

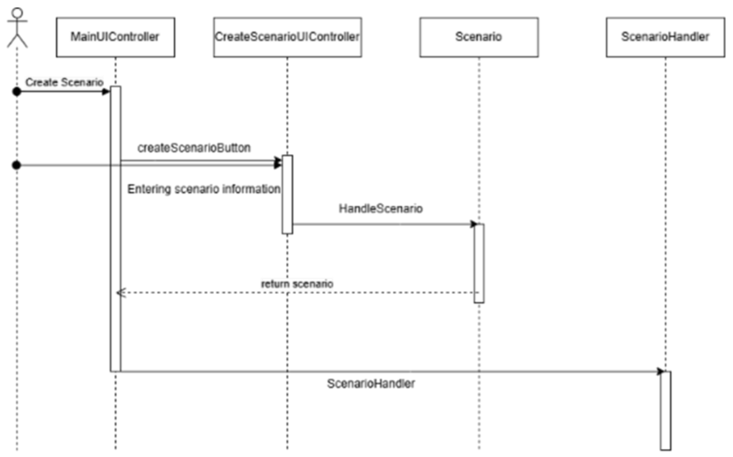

Figure 4 : Create Scenario

The sequence diagram shown in ***Figure 4*** illustrates the process of creating a scenario within the RTSG application.

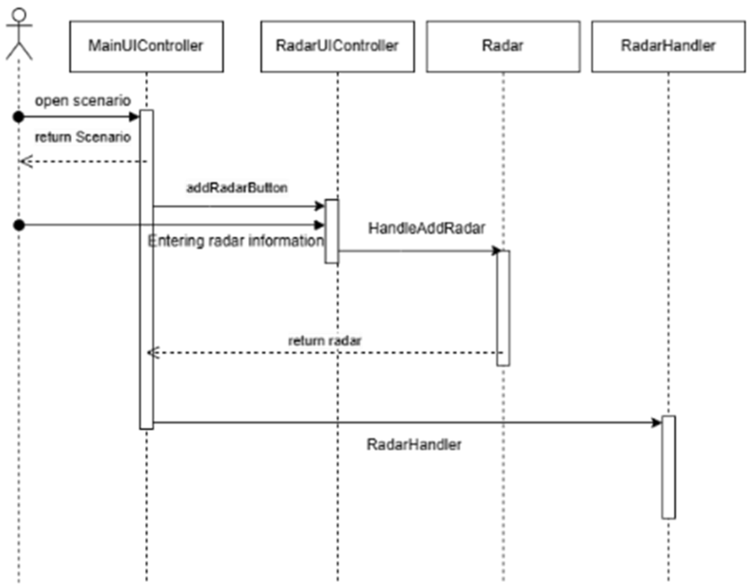

Figure 5 : Add Radar

The sequence diagram shown in ***Figure 5*** illustrates the process of adding a radar in the RTSG application.

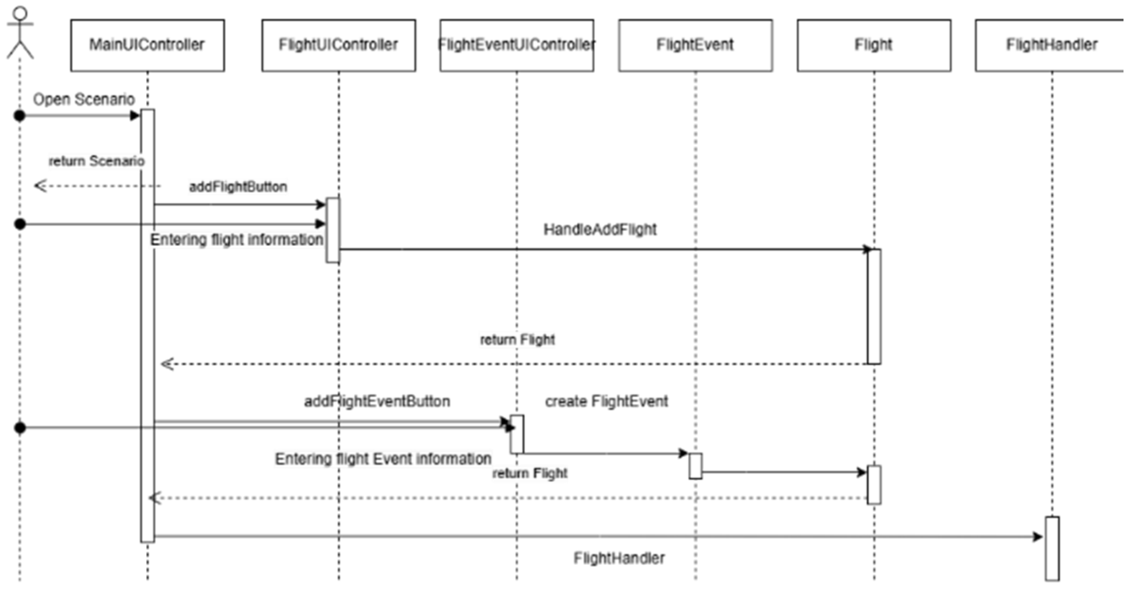

Figure 6 : Add Flight

The sequence diagram shown in ***Figure 6*** illustrates the process of adding a flight in the RTSG application.

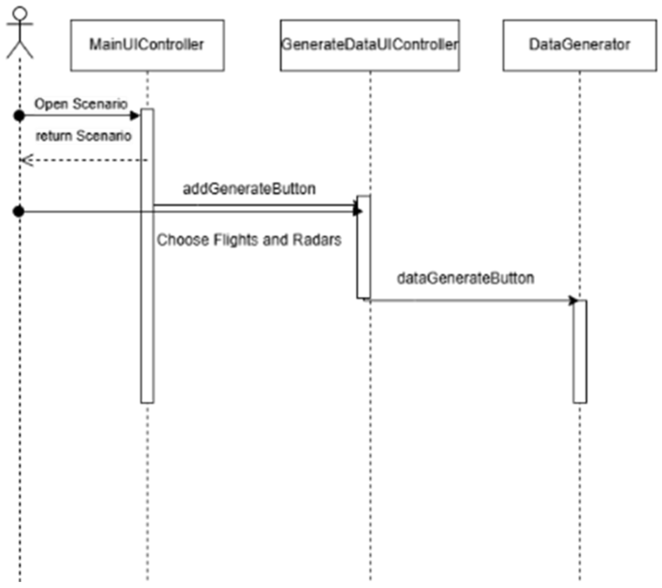

Figure 7 : Generate Data

The sequence diagram shown in ***Figure 7*** illustrates the process of generating data within the RTSG application.

4.2.2.3 Deployment View

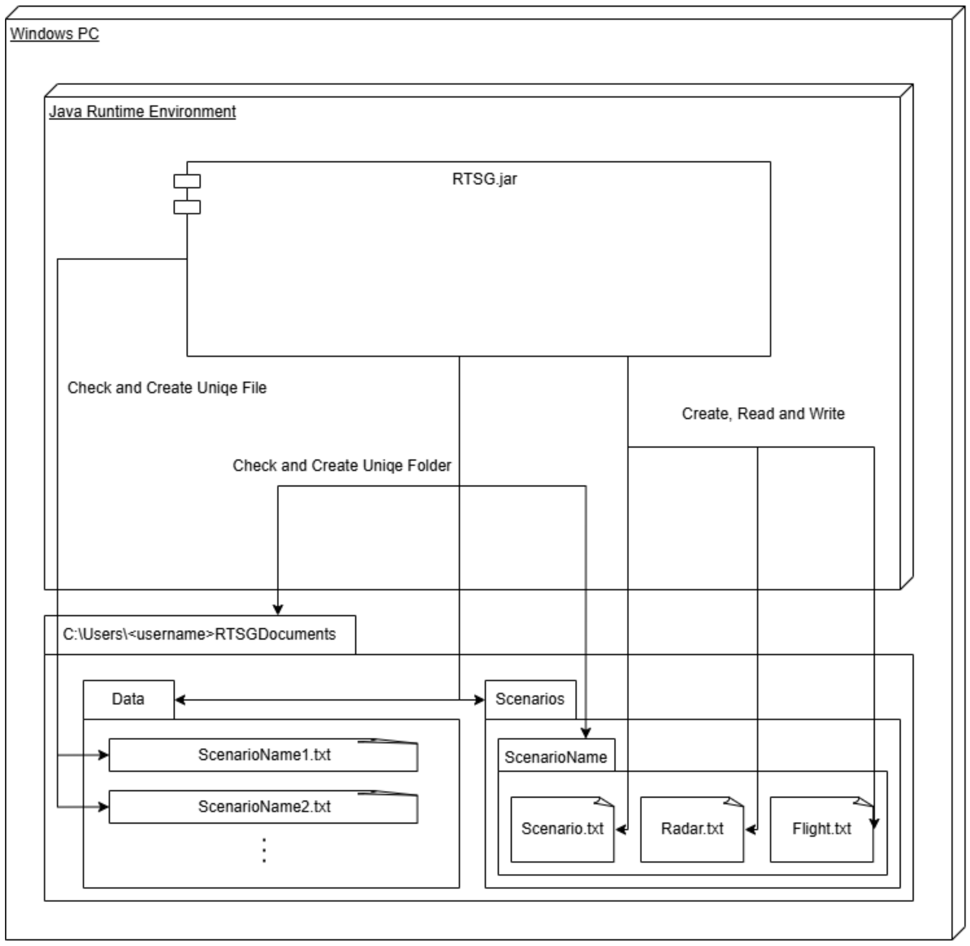

Figure 8 : Deployment Diagram

The deployment diagram shown in ***Figure 8*** depicts the physical distribution of the components in the RTSG project and how they are configured on the hardware. The deployment diagram is crucial for understanding the system's architecture from a hardware perspective, ensuring that the components are efficiently distributed and function correctly in a real-world setting.

4.3 Development

The development of the RTSG project was carried out with a strong emphasis on software engineering principles, ensuring a robust, maintainable, and scalable application. The design phase, guided by UML diagrams such as use case, class, and sequence diagrams, provided a structured approach for implementation. These diagrams helped define the system's functionality and relationships, forming the foundation for the development process.

The software architecture was designed based on the Singleton design pattern, which allowed for the controlled creation of critical components, ensuring only one instance was active where necessary. This design choice simplified the development process, minimized errors, and made debugging more manageable. The user interface was developed using JavaFX in combination with IntelliJ IDEA and Scene Builder, adhering to responsive design principles to ensure usability across various screen sizes and devices.

To manage the development lifecycle effectively, Jira was used for task management. Development was structured around software change requests (SCRs), each representing a specific feature or functionality. Tasks were implemented, committed, and pushed to a Git repository hosted on Bitbucket.

The application of tools such as Git and Bitbucket streamlined the workflow and maintained a well-documented version history. Each stage of the development process was accompanied by rigorous testing, ensuring that the implemented features met the required standards. The use of the Singleton pattern also contributed to reducing complexity during debugging and maintenance, further enhancing the system's reliability.

While the source code of the application cannot be shared due to confidentiality policies, it is securely stored within the company's database. However, the development methodology, supported by modern tools and UML diagrams, provides a comprehensive overview of the system's structure and implementation. By adhering to these principles and leveraging collaborative tools, the RTSG project achieved its objectives effectively, resulting in a well-designed and functional application.

4.4 Test

In this section, the testing procedures employed throughout the RTSG project are discussed. The testing process was critical to ensuring that the components and the overall system functioned as expected. The tests included unit tests, integration tests, and user interface (UI) tests.

Unit Testing:

Unit tests were used to verify individual methods and components of the system. Two specific examples of unit tests are:

Flight Progression Algorithm:

A unit test was written to ensure the flight progression algorithm accurately updated the flight data according to defined parameters such as time intervals and velocity.

Radar Coverage Validation:

A unit test was developed to verify whether radar systems correctly identified flights within their coverage areas. This test used the radar’s position and coverage radius to check whether a flight's location fell within the radar's defined range. The test ensured that the radar coverage logic was precise and reliable, a critical aspect for the proper functionality of the RTSG system.

Integration Testing:

Integration testing was carried out to check the interaction between various system components.

System Testing:

System testing involved the verification of the entire RTSG system, including its performance and functionality under different scenarios. This included:

Performance Testing:

The system was tested to ensure it could handle large volumes of radar and flight data without performance degradation.

Human-Computer Interaction (HCI) Testing:

UI tests were continuously performed to assess the user experience and ensure that the system was user-friendly and met the expectations of test engineers.

Alpha and Beta Testing:

In the final stages of development, alpha testing was conducted within the development team, where the system was tested for bugs and usability. Subsequently, beta testing was performed with the project manager and team leader, simulating real-world scenarios to identify and resolve potential issues before the final release.

Test Documentation:

Test cases were carefully documented and mapped to the project requirements in the company’s database. Due to confidentiality constraints, the detailed test documentation cannot be shared. However, these documented test cases played a crucial role in ensuring the system met all necessary specifications.

4.5 Evaluation

During Increment 1, the requirements and design phases were successfully completed. As the initial step of the project, the core functionalities of the RTSG system were defined, and design details were carefully addressed. However, observations and evaluations throughout the process revealed the need for certain improvements and additions.

In the data generation phase, the manual selection of radars and flights was initially required. However, it was realized that this step is unnecessary since each scenario already has its own radar and flight information. This will be addressed in Increment 2, where the data generation process will be automated to enhance usability and efficiency.

Additionally, it was determined that a category attribute needs to be added to flights. This feature will enable better organization and facilitate data management and analysis processes.

Regarding event management, a review highlighted the need to introduce a new event type called "no detection," alongside the existing "change speed" and "change heading" events. This enhancement will contribute to creating more realistic and comprehensive scenarios.

Lastly, it was identified that events require unique IDs assigned using a global counter. This approach will improve traceability and provide a more structured data management system.

In conclusion, while the requirements and design phases in Increment 1 were successfully accomplished, the identified development needs will be addressed in Increment 2. These changes and additions aim to make the system more efficient, user-friendly, and flexible.

Ch5: Increment 2

5.1 Requirements Update

In Increment 2, the requirements were revised based on the findings and evaluations from Increment 1. The following updates were identified and implemented to enhance system functionality and efficiency:

- The manual selection of radars and flights during the data generation phase was removed. Each scenario now automatically utilizes its own radar and flight information, eliminating redundancy.
- A category attribute was added to flights, allowing for better classification and management.
- A new event type, no detection, was introduced alongside the existing change speed and change heading events to improve scenario realism.
- Events were updated to include unique IDs assigned using a global counter for better traceability and management.

5.2 Design Update

The design phase was updated to incorporate the revised requirements:

- The UML diagrams were modified to reflect the removal of manual radar and flight selection during data generation.
- The flight class was updated to include a category attribute, and the event class was updated to include a global counter for assigning unique IDs.
- Sequence diagrams were revised to integrate the no detection event and ensure seamless functionality within the existing system.

5.3 Development

Development focused on implementing the updates and refining the system:

- The data generation module was redesigned to automatically utilize the radars and flights associated with each scenario, improving workflow and usability.
- The category attribute was added to the flight class, and methods were adjusted to handle this new property effectively.
- The new no detection event type was integrated into the event management system, alongside changes to accommodate the global counter for event IDs.
- The changes were committed and managed using Git and Bitbucket, ensuring version control and collaboration.

5.4 Test

Comprehensive testing was conducted to verify the updates:

- **Unit Testing:** Verified the automated radar and flight selection during data generation, the correct functionality of the category attribute, and the proper integration of the no detection event.
- **Integration Testing:** Ensured that the changes worked seamlessly with existing modules, including data generation, scenario execution, and event handling.
- **System Testing:** Validated the overall performance and usability of the system, confirming that the updates enhanced system efficiency and accuracy.
- **UI Testing:** Ensured that the new features were reflected appropriately in the user interface, maintaining a user-friendly design.

5.5 Evaluation

The Increment 2 updates successfully addressed the issues identified during Increment 1.

- Automating radar and flight selection reduced redundancy and improved the user experience.
- The addition of the category attribute provided a more structured approach to managing flights.
- The no detection event type added a new dimension to scenario creation, making the system more realistic.
- The implementation of global counter IDs for events improved traceability and data organization.

The system is now more efficient, flexible, and aligned with user requirements, setting a strong foundation for future increments.

Ch8: Impact of the Project & Compliance with the Constraints

8.1 Compliance with the Realistic Constraints

Your project may have some realistic constraints (cost, cost effectiveness, runtime and memory constraints, performance and so on). Some of these have already been mentioned in your SRS documents (I hope). Discuss here, whether these constraints are met or not. You may also discuss the new constraints you have encountered during the development and implementation (runtime problems, memory problems and so on). After the discussion, summarize your findings by filling out the appropriate boxes in the table below. Enter NA if an item does not apply to your project.

Table 8.1 Realistic Constraints & Conditions

Economic Factors	**Please specify/explain realistic constraints and conditions (type, use, amount, etc.)**
EXPENDITURES	
Computer	The computer used for software development and testing was provided by Ayesaş.
Other Devices	No additional devices were needed; the general devices provided by Ayesaş were sufficient.
Peripherals	Peripherals such as keyboards and mice were provided by Ayesaş.
Internet Connection	The internet connection was provided by Ayesaş.
Software	All necessary software and licenses for the project were provided by Ayesaş.
Textbook/Magazine/Support Material	There was no need for any additional books or support materials specific to the project.

Human Resources	The project team was provided by Ayesaş, and only in-house employees were involved.
Other	NA
FUNDING Sources	
University Resources	University resources were not used as the project was carried out at Ayesaş.
Project support (SANTEZ, TÜBİTAK, and so on)	No external support was obtained for the project.
Support by the Industry	The project was fully supported by Ayesaş.
Self-funded	No self-funding was used; the project was financed by the budget provided by Ayesaş.
Other sources	NA
OTHER CONSTRAINTS	
Memory	No memory constraints were encountered, and sufficient resources were provided.
Runtime efficiency	The runtime efficiency of the application met the targeted standards.
MANAGERIAL	
Schedule (time)	The project was completed within the specified timeline.

8.2 Impact of the Project

Here, discuss the impact of your project on the society, health and environment. After a short discussion on these enter your views and points into the table below as appropriate. If an item is not applied to your project just say NA.

Table 8.2 Impact Assessment Report

Professional/Ethical Issues	**Please specify/explain (existence of items, violation of items, awareness about items)**
ETHICS/IT Law	Are there any applicable laws or legislation that will be relevant to the use and/or the construction of the system you are building? How have you addressed them?
Copyright in copying multimedia (sound, video, text)	All multimedia used in the RTSG project is properly licensed or created by the development team to avoid copyright infringement.
Use of licensed software	The RTSG project uses licensed software in accordance with the relevant software licenses to ensure compliance and avoid legal violations.
Data Privacy	The project adheres to data privacy laws by ensuring that personal or sensitive data is protected through encryption and secure handling practices.
Use of patented products/ideas	Any patented products or ideas used in the project are incorporated with proper authorization or licensing to avoid patent infringement.
IT Laws in Turkey (5661 and others)	The RTSG project complies with Turkish IT laws, including Law No. 5661, by ensuring that all online and data

	processing activities are in line with national regulations.
IT Laws - International	The project adheres to international IT laws, such as GDPR, ensuring that it meets global standards for data protection and cybersecurity.
Plagiarism	The RTSG project has been developed with original code and design, and proper citations and credits are given for any third-party contributions to avoid plagiarism.
PROFESSIONAL	Liability/Sustainability Issues – What is the potential for liability either to yourself or your customer or users of the system, if it is misused or has flaws? Financial Impact – Have the financial costs of deploying and supporting the system been fully evaluated for yourself, the customer, users, and society as a whole?
Sustainability (use of Licensed and/or open source code)	The RTSG project ensures sustainability by adhering to appropriate licensing terms and using open-source components where applicable.
Maintenance	The system is designed for easy maintenance, with regular updates and user feedback incorporated to ensure long-term reliability.
Liability	In case of misuse or flaws, liability may fall on the developers or the organization, depending on the terms of use and the context of deployment.
Financial impact/Manufacturability	The financial impact has been evaluated, ensuring cost-effective deployment and long-term support without significant financial burden on the customer or users.

SOCIAL/POLITICAL/ ENVIRONMENTAL	Societal Issues – What is the potential for this system to be beneficial or detrimental to society as a whole? What have you done to address the potentially detrimental use? Are there any side-effects on the environment? Are there any potential political implications of the use of your system? What impact will this system have on the intended user community? Have you taken steps to safeguard the interests of that community?
Political impact	NA
Impact on health	NA
Gambling	NA
Pornography	NA
Equal Access/equity	NA
Environmental impacts (energy, carbon footprint and so on)	NA
Technology acceptance & Human/Business psychology	NA
Security issues	NA
PROFESSIONAL (CODES, STANDARDS, FRAMEWORKS)	What technical standards are relevant to the software system you have built? How have you ensured conformance? These could be government standards, industry standards, or even just conventions that are followed in the market for your system. Be sure to clearly identify what type of standard you are talking about and who is the relevant authority for the standard.

IEEE	IEEE 12207
ISO	ISO 27001
ANSI	NA
TSE	NA
ITIL/COBIT	NA
OTHER	NA

Ch9: Conclusions

The Radar and Track Scenario Generator (RTSG) project successfully addressed the challenge of generating realistic radar and moving object data for testing track fusion algorithms. By utilizing an incremental development approach, the project has significantly improved its core functionalities over two key releases. The first increment laid the foundation by allowing users to create and manage radar and track scenarios, while the second increment refined the user experience and system performance, incorporating user feedback to optimize features.

The RTSG project's contribution to the field is notable in its ability to simulate realistic radar and flight scenarios, which are essential for evaluating the performance of track fusion algorithms. By generating synthetic data based on predefined scenarios, the project provides a reliable platform for testing these algorithms under controlled but realistic conditions. This capability is critical for industries such as air traffic control, military operations, and civilian tracking applications, where the accuracy and reliability of tracking systems are paramount.

The integration of user feedback throughout the project development process ensured that the final product met the needs of its target users, enhancing both functionality and usability. As a result, RTSG serves as an important tool for improving the efficiency and accuracy of radar systems, ultimately contributing to safer and more reliable systems in a variety of fields. The project not only advances technical capabilities in radar and track scenario generation but also demonstrates the value of incremental development and user-centered design in producing robust software solutions.

Appendix A

The detailed program listings, data, and additional information related to the RTSG project are available in the company’s internal database.

References

[1] Richards, M. A., Scheer, J. A., & Holm, W. A. (2010). Principles of Modern Radar: Basic Principles. SciTech Publishing.

[2] Barton, D. K., & Leonov, S. A. (2013). Radar Technology Encyclopedia. Artech House.

[3] Blackman, S. S., & Popoli, R. (1999). Design and Analysis of Modern Tracking Systems. Artech House

[4] Hall, D. L., & Llinas, J. (2001). Handbook of Multisensor Data Fusion: Theory and Practice. CRC Press.

[5] Law, A. M. (2014). Simulation Modeling and Analysis. McGraw-Hill Education.
[6] Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2010). Discrete-Event System Simulation. Prentice Hall.

[7] Skolnik, M. I. (2008). Radar Handbook. McGraw-Hill Education.

[8] Mahafza, B. R. (2016). Radar Systems Analysis and Design Using MATLAB. CRC Press.