sUAS Safety Case Tool

Final Report

Team Members:
Brady Bargren
Blake Bryan
Benjamin Kelly
Gabriel Perez
Gautham Suresh

Client: Myra Cohen
Course: Senior Design
Instructor: Simanta Mitra
TA: Shaiqur Rahman

Iowa State University
December 17, 2024

Client Letter

[Client letter expressing thoughts/feelings about the team's work will be attached here.]

Contents

Chapter 1

Requirements

1.1 Functional Requirements

- Parse XML models to generate dynamic questionnaires tailored to specific sUAS configurations. The XML models will be dynamically parsed to handle complex nested structures with AND/OR logic, ensuring that the generated questionnaires adapt to user-specific flight scenarios.
- Generate safety cases in YAML format with corresponding graphical representations using Goal Structuring Notation (GSN). The safety cases visually represent safety arguments, including goals, strategies, evidence, and risk indicators.
- Support dynamic risk assessment based on user inputs. A risk scoring algorithm will evaluate responses and categorize the overall flight risk as low, moderate, or high.
- Store user responses in a database for future retrieval and regulatory compliance. Responses will be stored in MongoDB to allow users to retrieve historical data and maintain compliance records for regulatory purposes.
- Fetch real-time airspace updates using NOTAMs and incorporate them into the risk assessment process, providing pilots with critical situational awareness.

1.2 Non-Functional Requirements

- Ensure user interface responsiveness and ease of use. The React-based frontend is optimized for clarity and usability, featuring intuitive navigation and dynamically generated forms.
- Maintain data integrity and secure storage in MongoDB. All data must be securely stored and encrypted to prevent unauthorized access or data loss.
- Provide compatibility with existing regulatory standards, ensuring that safety cases and risk assessment outputs align with FAA guidelines and other regulatory bodies.
- Allow for real-time updates, such as integrating NOTAMs data seamlessly into the application to enhance decision-making during pre-flight safety checks.

- Ensure high performance and scalability to support at least 100 simultaneous users with an average safety case generation time of less than 5 seconds.
- Design the system to support deployment on cloud platforms like AWS or Azure, ensuring high availability and reliability.

1.3 Additional System Attributes

1.3.1 Reliability

- Achieve 99.9% uptime for all critical features, including safety case generation and data retrieval.

1.3.2 Availability

- Ensure the system is accessible 24/7, even under high user loads.

1.3.3 Maintainability

- Utilize a modular codebase with well-documented components to facilitate future enhancements and maintenance.

1.3.4 Portability

- Design the application for compatibility with major cloud platforms and local deployments, ensuring adaptability to different environments.

1.4 Constraints and Assumptions

- Real-time NOTAM data retrieval is dependent on external API access and may face occasional delays or limitations due to API constraints.
- Users are assumed to have access to modern devices with HTML5-compliant browsers and a stable internet connection.
- The backend system relies on Node.js and MongoDB being properly installed and configured on the server.
- Safety cases must adhere to GSN standards, and any changes to these standards may require updates to the system.

Chapter 2

Design

2.1 System Architecture

The project follows a modular architecture, enabling efficient functionality and maintainability:

- **Frontend**: Built with React and TypeScript, the user interface dynamically generates questionnaires, displays safety cases, and provides visual feedback for risk assessments.
- **Backend**: A Node.js server that manages XML parsing, processes user responses, generates safety cases, and calculates flight risk.
- **Database**: MongoDB handles persistent storage for user responses, XML models, and safety case data, supporting scalability and secure data handling.
- **Safety Case Generator**: Scripts process XML/JSON inputs, convert them to YAML format, and generate visual representations using Goal Structuring Notation (GSN).

2.2 API Design

The system employs RESTful APIs to manage data interactions between the frontend, backend, and external services:

2.2.1 Endpoints

- **GET /model**: Fetches a dynamically generated questionnaire based on the XML model.
- **POST /submit**: Accepts user responses, generates safety cases, and stores data for future use.
- **GET /responses**: Retrieves previously submitted user responses from the database.
- **GET /safety-case-data**: Provides detailed data for generated safety cases, including their risk classification.

- **GET /notams**: Fetches Notices to Air Missions (NOTAM) data for a specified location to enhance situational awareness.

2.3 Design Rationale

2.3.1 Development Standards & Practices Used

- **Frontend Standards**: Developed using React with TypeScript for modular and type-safe component design, and Bootstrap for responsive and accessible layouts.
- **Backend Practices**: Used Node.js with Express for scalable and efficient API design. MongoDB was chosen for its flexible schema and support for complex query operations.
- **Safety Case Standards**: Generated safety cases follow Goal Structuring Notation (GSN), ensuring compatibility with industry expectations.
- **Risk Assessment Integration**: Implemented dynamic algorithms to analyze user inputs and determine risk levels.

2.3.2 Design Decisions

- **Why React for the Frontend?** React allows the development of a highly dynamic and interactive user interface. Its component-based design simplifies updates and maintenance.
- **Why Node.js for the Backend?** Node.js ensures asynchronous, event-driven server-side operations, making it ideal for handling multiple concurrent users with minimal latency.
- **Why MongoDB for the Database?** MongoDB's schema-less design aligns with the evolving requirements of the application, especially for storing XML models, user responses, and YAML safety cases.
- **Why GSN for Safety Cases?** GSN visually represents safety arguments, allowing stakeholders to easily interpret dependencies, risk factors, and evidence.

2.3.3 Addressing Challenges

Challenge 1: XML Parsing Complexity

- **Problem**: XML models include nested structures with AND/OR logic, requiring recursive processing.
- **Solution**: Developed a custom parser utilizing tree traversal algorithms to handle complex dependencies and hierarchical structures efficiently.

Challenge 2: Dynamic Risk Calculation

- **Problem**: Risk evaluation depends on multiple variables with complex interdependencies.

- **Solution**: Implemented a dynamic scoring algorithm that adjusts risk levels based on user responses, flight configurations, and environmental data.

Challenge 3: YAML and GSN Compatibility

- **Problem**: Existing tools for GSN diagram generation were incompatible with YAML-based safety cases.
- **Solution**: Developed scripts to translate YAML safety case data into GSN-compatible structures for accurate visual representation.

2.4 Components

2.4.1 Frontend Application

- Built using React and TypeScript for a modern, responsive user interface.
- Handles dynamic form generation and safety case visualization.
- Integrates Bootstrap for consistent styling and enhanced accessibility.

2.4.2 Backend Server

- Developed using Node.js and Express for scalable and efficient API endpoints.
- Implements algorithms for XML parsing, risk assessment, and safety case generation.
- Interfaces with MongoDB for data storage and retrieval.

2.4.3 Database (MongoDB)

- Stores XML models, user responses, and generated safety cases in a structured format.
- Ensures secure data storage with access controls and encryption mechanisms.
- Supports scalability for concurrent users and high query loads.

2.4.4 Safety Case Generator

- Converts XML/JSON data into YAML safety case files.
- Generates Goal Structuring Notation (GSN) diagrams for visual representation of safety arguments.
- Supports real-time updates and visual feedback based on user inputs.

2.5 Data Flow

- User inputs are captured by the React frontend and sent to the backend via API calls.
- The backend processes responses, stores data in MongoDB, and generates YAML safety cases.
- GSN diagrams are rendered dynamically from YAML data and displayed in the frontend.
- External data (e.g., NOTAMs) is retrieved through backend APIs and incorporated into risk assessments.

2.6 Architectural Diagram

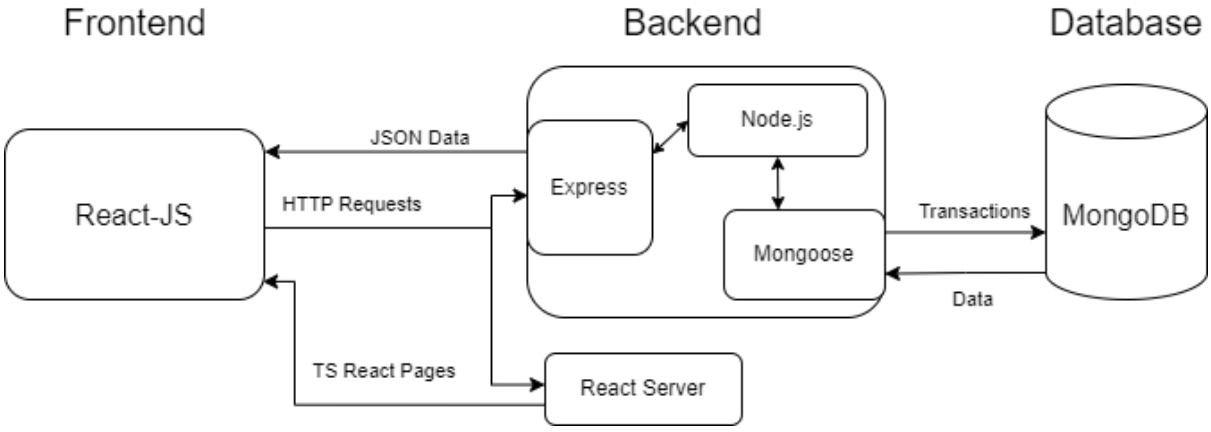

Figure 2.1: System Architecture for the sUAS Safety Case Tool

Chapter 3

Work Done

3.1 Learning New Technologies

The project required the team to adopt and learn various new tools, technologies, and frameworks to meet the requirements efficiently:

- **MongoDB**: Utilized for database management to store and retrieve user responses.
- **React.js with TypeScript**: Leveraged to build a dynamic, responsive frontend interface.
- **xml2js Library**: Enabled parsing of complex XML models for generating dynamic questionnaires.
- **js-yaml Library**: Facilitated conversion of data into YAML format for safety case generation.
- **Goal Structuring Notation (GSN)**: Adopted for creating graphical safety case representations that are visually interpretable.
- **Dynamic SVG Manipulation**: Integrated risk assessment outputs into visual safety case diagrams using color-coded risk indicators.
- **Node.js with Express**: Provided a scalable backend framework for managing API interactions and data processing.
- **Jest**: Used for automated testing of backend components to ensure system reliability.

3.2 Team Member Contributions

The following outlines the roles and significant contributions of each team member during the project's development:

3.2.1 Brady Bargren (Project Manager / Backend Developer)

- Coordinated overall project efforts, managed deadlines, and maintained the task schedule.
- Developed YAML generation scripts and integrated MongoDB into the backend for secure data storage.
- Implemented the risk assessment algorithm and designed the colorization module for dynamic SVG diagrams.
- Created an installer batch file to streamline dependency installations and simplify the setup process for future developers.
- Ensured proper management of timestamped files for chronological ordering and accurate safety case data handling.

3.2.2 Benjamin Kelly (Frontend Developer)

- Designed and implemented a dynamic questionnaire interface using React and TypeScript.
- Managed state handling, form validation, and user experience optimizations, including cookie-based form state persistence.
- Developed the frontend functionality for displaying generated safety case diagrams and survey results.
- Enabled the visualization of historical user responses to support iterative safety case reviews.

3.2.3 Gabriel Perez (Backend Developer)

- Parsed the XML model to dynamically generate questions based on sUAS configurations, addressing complex AND/OR/ALT nodes through recursive tree traversal algorithms.
- Managed the argument parser pipeline to process and structure user inputs into JSON/YAML formats.
- Ensured proper data flow consistency between the frontend and backend.
- Contributed to debugging and resolving issues in YAML safety case generation.

3.2.4 Gautham Suresh (Backend Developer)

- Managed interactions between the backend and MongoDB to store, retrieve, and maintain data integrity.
- Refined and modularized backend APIs for improved scalability and maintainability.
- Assisted in designing the risk level calculator and contributed to the argument parsing pipeline for GSN diagram generation.

3.2.5 Blake Bryan (Frontend/Backend Developer)

- Supported the development of system architecture diagrams to align team efforts on project structure.
- Implemented functionality for rendering and integrating generated safety case images into the frontend.
- Worked on backend and frontend to convert feature model xml into React components that make up the form users interact with.
- Addressed challenges in handling timestamped files and integrated their display into the frontend for enhanced user tracking.

3.3 Challenges Faced

3.3.1 Backend Challenges

- **Complex XML Parsing**: Addressing nested AND/OR logic in XML models required iterative testing and debugging.
- **YAML Generation**: Ensuring the generated YAML accurately represented diverse sUAS configurations.
- **Timestamp Management**: Developing a reliable process for managing timestamped files to maintain data integrity and display order.
- **NOTAM API Access**: Overcoming restrictions and CAPTCHA-related issues while retrieving real-time airspace data.

3.3.2 Frontend Challenges

- **Data Management**: Ensuring smooth state transitions and proper synchronization of dynamic form inputs and backend responses.
- **Image Rendering**: Displaying generated safety case diagrams asynchronously within the user interface.
- **API Consistency**: Adjusting frontend components to accommodate changes in backend API structures.

Chapter 4

Results Achieved

4.1 Overview of Results

The sUAS Safety Case Tool successfully achieved its primary goals of automating safety case generation, integrating risk assessment mechanisms, and delivering a robust, user-friendly interface. The following results highlight the achievements made during the project:

- **Dynamic Questionnaire Generation**: Successfully implemented a dynamic questionnaire system by parsing XML-based flight configuration models. The questionnaire adapts to specific sUAS configurations, ensuring tailored safety case generation.
- **Safety Case Generation in YAML and GSN Diagrams**: Developed functionality to convert user inputs into YAML-based safety cases and visually represent them as Goal Structuring Notation (GSN) diagrams. These diagrams include color-coded risk levels to enhance user understanding.
- **Risk Assessment Integration**: Implemented a risk assessment algorithm to evaluate user inputs and categorize overall flight risks as low, moderate, or high. The algorithm dynamically adapts to user responses and generates corresponding visual indicators in the safety case diagrams.
- **Integration with NOTAMs**: Initiated integration with Notice to Air Missions (NOTAM) data, providing users with real-time updates on airspace restrictions and conditions. This functionality enhances situational awareness for pre-flight safety checks.
- **User-Friendly Interface**: Delivered a React-based frontend interface that dynamically generates questionnaires, displays safety case results, and allows users to access historical safety cases. Features such as form state management and intuitive navigation ensure an enhanced user experience.
- **Pruning Functionality**: Developed pruners for both YAML and JSON-based safety cases, enabling users to focus on problematic nodes by filtering irrelevant data. This functionality improves clarity and usability of safety case diagrams.

- **Historical Data Management**: Implemented a system for storing and retrieving historical safety case data, allowing users to review previous assessments and track changes over time.

4.2 Key Milestones Achieved

The team successfully met several critical milestones during the development process:

- **Dynamic Questionnaire System**: Designed and implemented a robust system that parses complex XML flight models to generate dynamic, adaptive questionnaires.
- **Safety Case Visualization**: Delivered a tool that converts structured data into visually interpretable GSN diagrams, integrating risk assessment outputs with visual color coding for clarity.
- **Backend-Frontend Integration**: Achieved seamless communication between the backend server and the frontend interface, ensuring consistent data flow and synchronization.
- **Real-Time Data Integration**: Began integrating real-time NOTAM data into the tool, setting the foundation for enhanced situational awareness.
- **Installer Batch File Development**: Streamlined project setup through the creation of an installer batch file that automates dependency installation for future developers.
- **Enhanced Risk Analysis**: Incorporated a robust risk assessment mechanism that dynamically evaluates user responses and reflects them in the safety case results.
- **SVG Colorization Module**: Developed an SVG-based visualization module that dynamically colorizes GSN diagrams based on risk levels, aiding in quick decision-making.

4.3 Impact and Benefits

The achievements of the project directly address the challenges faced by sUAS pilots in conducting pre-flight safety assessments:

- **Standardization of Safety Cases**: The tool introduces a standardized approach to generating safety cases, reducing reliance on manual processes and ensuring consistency across different scenarios.
- **Improved Decision-Making**: The integration of dynamic risk assessments and visual safety diagrams empowers pilots to make informed decisions about flight safety.
- **Enhanced User Experience**: The user-friendly interface, coupled with features like historical data retrieval and real-time NOTAM integration, simplifies safety case generation for pilots.

- **Regulatory Compliance Support**: The ability to store and retrieve historical safety case data ensures compliance with potential regulatory requirements.

4.4 Remaining Work and Future Enhancements

Although the project successfully met its goals, there are opportunities for further enhancements:

- **Expanded Risk Algorithm**: Integrate additional operational and environmental factors into the risk assessment algorithm for more comprehensive safety evaluations.
- **Enhanced NOTAM Integration**: Finalize real-time NOTAM data integration and incorporate it into the risk assessment process.
- **Simplified Safety Case Diagrams**: Introduce more user-focused pruning functionalities to simplify diagrams further, emphasizing critical risk nodes.
- **Downloadable Reports**: Enable users to download safety case results in standardized formats (PDF, YAML) for record-keeping and regulatory compliance.
- **Performance Optimization**: Optimize system performance to support large-scale deployments and reduce latency in generating safety cases.

Appendix A

Supporting Documents

A.1 Software Requirements

The software requirements for the sUAS Safety Case Tool can be found in the document linked below:

sUAS_Software_Requirements.pdf

This document includes:

- Functional and non-functional requirements for the application.
- Definitions, acronyms, and references.
- Detailed system scope, constraints, and assumptions.

A.2 Design Rationale

The design rationale for the project is available in the document linked below:

sUAS_Architecture.pdf

This document covers:

- Design decisions and justifications.
- Data decomposition and flow.
- Architectural diagrams and detailed system components.

Appendix B

Demo Slides

Demo 1 Slides

Demo 2 Slides

Demo 3 Slides