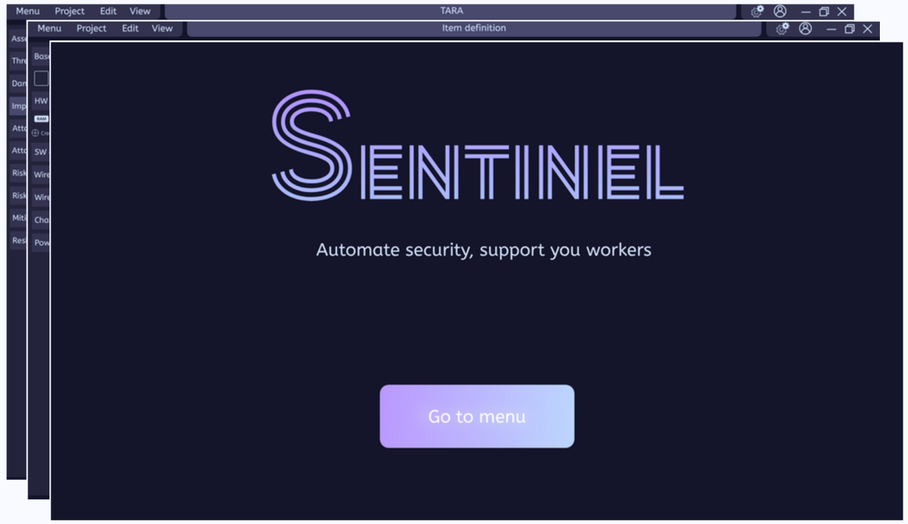
C++/Qt

Antonin GERMAIN

PERSONAL PROJECT



Sentinel - Application in Design

Table of contents

Problem Statement & Target Users

Context & Objective

This project aims to develop a **modern** and **intuitive tool** for cybersecurity risk analysis based on the **TARA** (Threat Analysis and Risk Assessment) methodology of **ISO21434** (automotive cybersecurity standard).

The tool allows users to visually create embedded **system architectures** by placing and connecting **predefined components** (e.g., UDS interfaces, JTAG ports, communication modules, CPU, memory, etc.).

Based on the user's design, the application **automatically** generates a **preliminary TARA**, to be further **refined** and **validated** by a cybersecurity **specialist**.

Identified Problem

Current TARA tools — like Medini Analyze — are widely used in the industry, but they often:

- Have outdated and cluttered interfaces,
- Are not beginner-friendly,
- Lack interactive or graphical system modeling capabilities.

Proposed Solution

This application aims to bring a **fresh**, **user-friendly**, and **automated** alternative that simplifies the modeling phase and accelerates early risk assessments.

Future versions will incorporate AI-based assistance and Generative AI to automatically generate initial archectures based on provided external inputs, enabling fully automated preliminary TARA generation.

Key Features

01 **Graphical Architecture Editor** Design embedded system components (e.g., ECUs) using a modern and intuitive drag-and-drop interface.

02 **Custom Component Library** Edit custom reusable components, stored in personal or shared libraries. Each component can be linked to multiple TARA steps.

03 **Automated TARA Report Generation** Automatically generate a TARA report based on the item defined system architecture and outputted risk analysis.

04 **Project Save & Load** Save your progress and reload complete analysis projects.

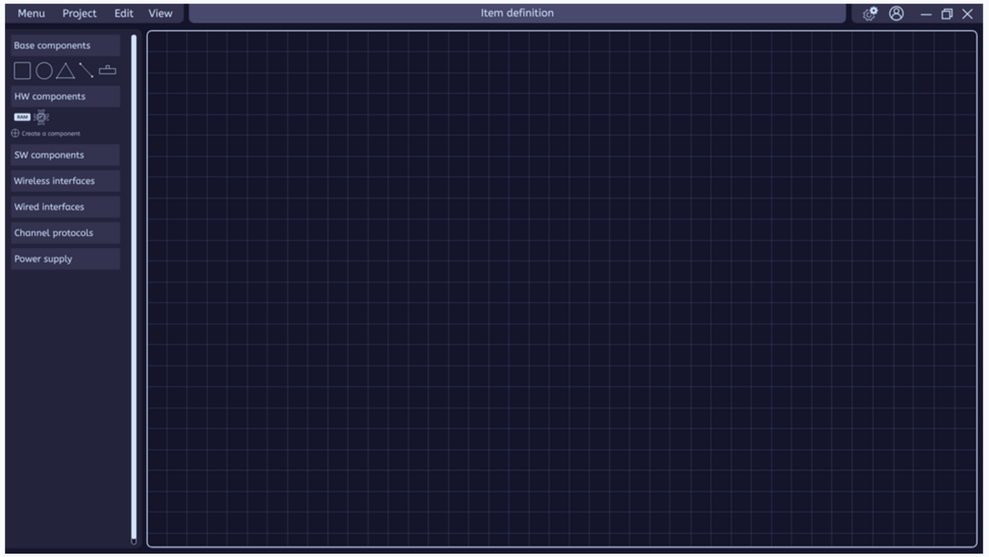
05 **Milestone Tracking** Update and adapt your TARA according to project milestones, with the ability to create custom milestones.

Full set of ~20 mockups available upon request

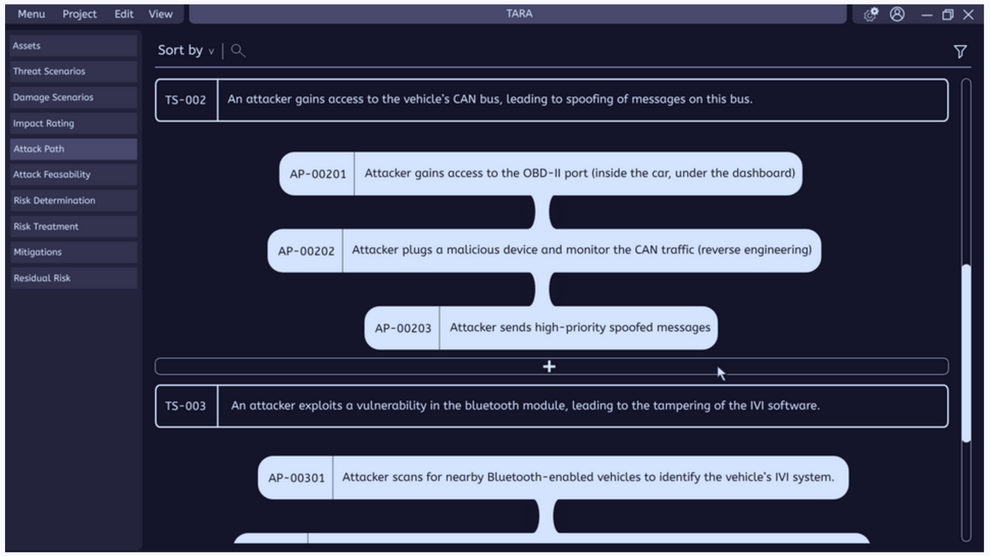
UI Mockups

Project Menu

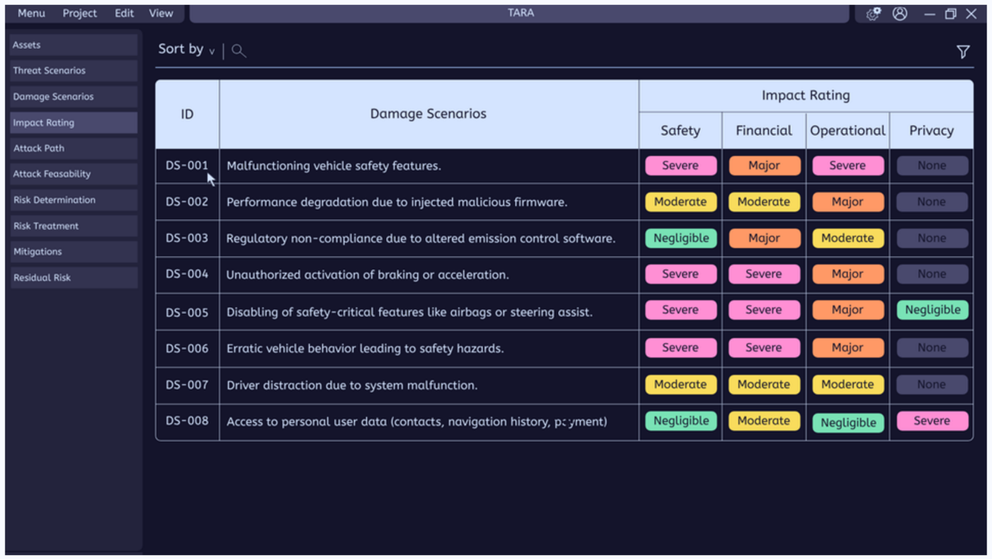
Diagram Editor



Attack Path

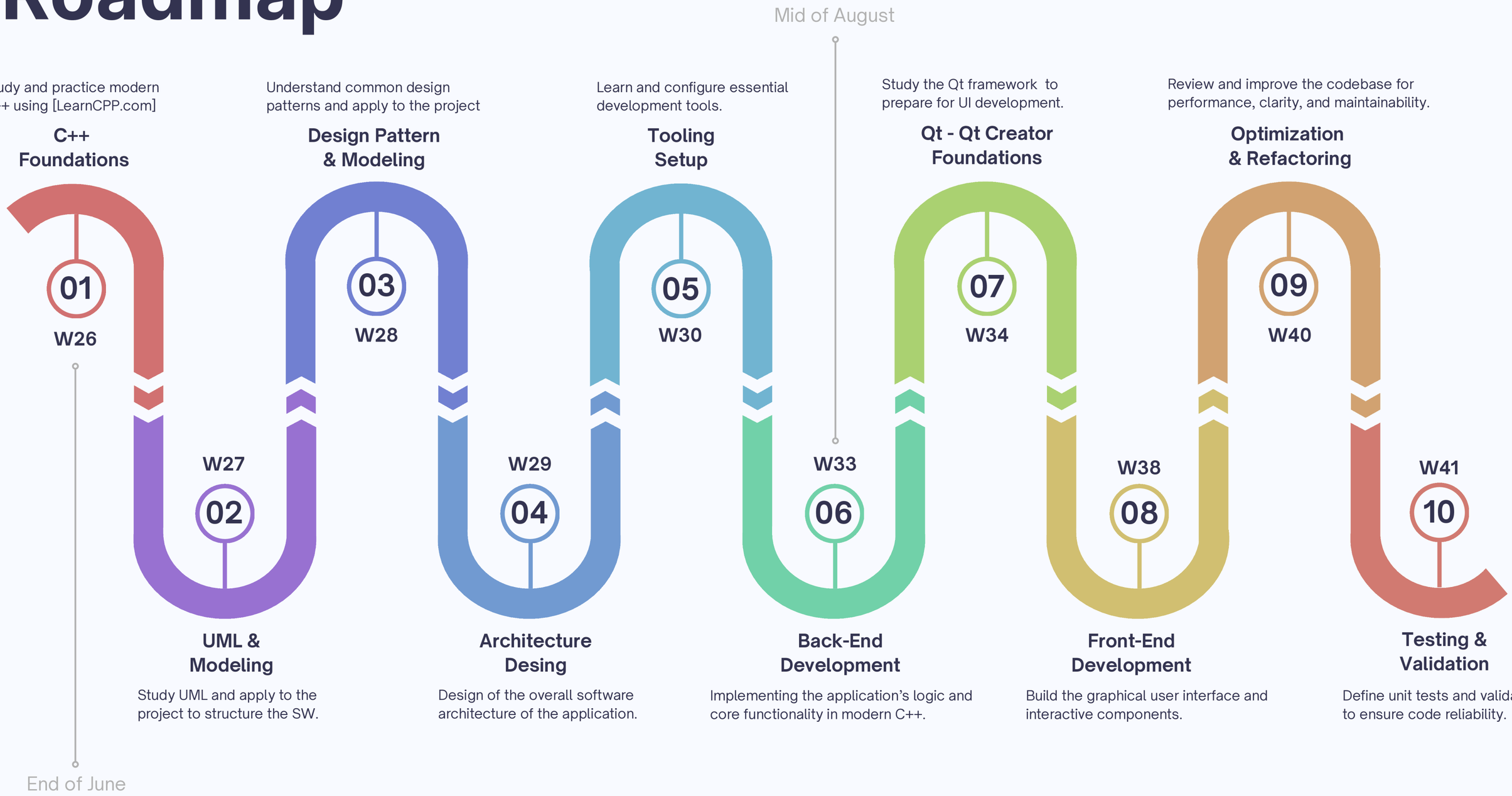


Impact Value



ID	Damage Scenarios	Impact Rating			
		Safety	Financial	Operational	Privacy
DS-001	Malfunctioning vehicle safety features.	Severe	Major	Severe	None
DS-002	Performance degradation due to injected malicious firmware.	Moderate	Moderate	Major	None
DS-003	Regulatory non-compliance due to altered emission control software.	Negligible	Major	Moderate	None
DS-004	Unauthorized activation of braking or acceleration.	Severe	Severe	Major	None
DS-005	Disabling of safety-critical features like airbags or steering assist.	Severe	Severe	Major	Negligible
DS-006	Erratic vehicle behavior leading to safety hazards.	Severe	Severe	Major	None
DS-007	Driver distraction due to system malfunction.	Moderate	Moderate	Moderate	None
DS-008	Access to personal user data (contacts, navigation history, p:yment)	Negligible	Moderate	Negligible	Severe

Roadmap


Mid of August
Study and practice modern C++ using [LearnCPP.com]
C++ Foundations
01
W26
End of June
W27
02
UML & Modeling
Study UML and apply to the project to structure the SW.
Understand common design patterns and apply to the project
Design Pattern & Modeling
03
W28
W29
04
Architecture Desing
Design of the overall software architecture of the application.
Learn and configure essential development tools.
Tooling Setup
05
W30
W33
06
Back-End Development
Implementing the application’s logic and core functionality in modern C++.
Study the Qt framework to prepare for UI development.
Qt - Qt Creator Foundations
07
W34
W38
08
Front-End Development
Build the graphical user interface and interactive components.
Review and improve the codebase for performance, clarity, and maintainability.
Optimization & Refactoring
09
W40
W41
10
Testing & Validation
Define unit tests and validation to ensure code reliability.

Technical Stack

Build & Dependency Management

CMake – Cross-platform build system for managing compilation, linking, and configurations.

Tooling & Environment

Visual Studio Code – Full-featured IDE for C++ development.
Qt Creator – Lightweight, Qt-native IDE with excellent integration for GUI development.
Docker – For environment consistency, reproducibility, and future deployment scenarios.
Git – Version control system used for tracking progress and collaboration.

Code Quality & Testing

Google Test – Writing and running unit tests in C++.
Clang-Tidy – Static analysis tool for detecting bugs, performance issues, and code style problems.
Clang-Format – Enforces consistent code formatting automatically.
Doxygen – Documentation generator to produce technical docs from annotated C++ source code.

Language Standards

C++11 / C++14 / C++17 / C++20
Qt 6.6+
UML 2.5
SQLite 3.x