Team Project: Developing and testing a software-based prototype of an AED

Textual explanation

Team 58 (3):

Evan Moore

William Trinh

Ziyang Ling

Explanation:

The document provides a comprehensive explanation of a team project focused on developing and testing a software-based prototype of an Automated External Defibrillator (AED).

State Pattern

The state pattern was used because the program's behaviour changed depending on which step of the process the AED was in. We used the approach of creating separate classes for each of the states and having them inherit a single abstract state class. State-specific logic is moved into their corresponding classes to keep the code organized. The current state is kept track of in the MainWindow class and the state is changed by calling changeState() on an instance of MainWindow and passing a new instance of a state.

The state classes have two special methods called execute() and initialize(). execute() is the main method that's called by the mainWindow when the state changes. When used with the IntermediateState class, the method can change behaviour depending on its current step. initialize() is similar to a constructor method and is called right after the state instance is initialized. The main purpose of the initialize method is to call the execute() method (if used). The reason why the method exists is that problems arise when calling execute() on one state while in another state.

Observer Pattern

The observer pattern is incorporated using Qt's signals and slots. We used it to subscribe to events emitted by buttons, combo boxes, spin boxes, etc. We've also defined new signals inside of the MainWindow to notify subscribers of changes to its state.
To accommodate the design of the software for a graphical user interface (GUI) that resembles the AED Plus's display, our team's design documentation and ideas are the following:

The design includes User Interface Design, Cardiac Arrhythmia Detection, Real-Time CPR Feedback, Visual Prompts, User Interaction and Simulated Scenarios.

The document details the design and functionality of a software-based prototype Automated External Defibrillator (AED).

Powered Off State:	The AED is turned off.
Self-Test State:	The AED performs a self-test to ensure its functioning correctly with device status indicators (e.g., battery, electrode placement).
Check Responsiveness State:	Instruct the operator to check if the patient is responsive.
Call For Help State:	Instruct the user to call for emergency assistance.
Attach Defibrillator Pads State:	This state indicates the need to attach defibrillator pads to the patient's chest.
Analyzing State:	The AED analyzes the patient's heart rhythm to determine if a shock is needed with Cardiac arrhythmia diagnosis results (shockable-non-shockable). Include a display panel to show the simulated ECG diagram and device status.
Perform CPR State:	Guiding the operator to perform CPR on the patient with Real-time CPR feedback. Also, simulate the measurement and assessment of chest compressions And loop back to Analyzing State until the end program cases are active to power off the program.
Powered Off State:	The AED is turned off.

End Program Spec.

1) Default: The program keeps looping between analyzing state and CPR state
2) CPR Revives Patient: At the end of the CPR state, it will print a message saying "CPR revived patient"
3) Shock Revives Patient: At the end of analyzing state, it will print a message saying "Shock revived patient"
4) Patient dies: Keeps looping between analyzing state and CPR state
5) EMS arrives: The program prints out a message saying "EMS arrives" AED powers off

The demo video can be found here. (https://youtu.be/zc1WP2dcdPA)