User Needs and Requirements

Code Critiquer System for the C Language and Embedded C

sdmay25-23:
James Joseph
Samuel Lickteig
Alix Noble
Andrew Sand
Owen Sauser

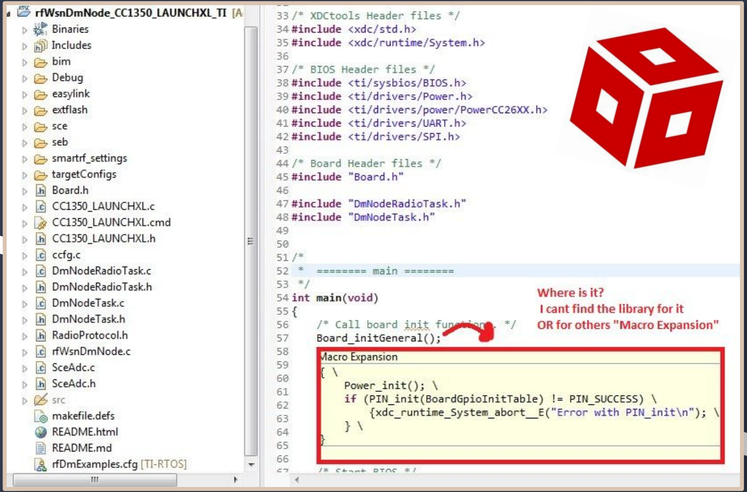
Code Critiquer System for the C Language and Embedded C

Project Overview

- Current state of project (continuation of sdmay24-34) is a web-based critiquer tool
  - Students upload C files to tool
  - Files are statically analyzed to search for antipatterns
  - Tool generates feedback
  - Students use feedback to improve skills
- Will modify current system and/or develop new prototypes
- Ideally tailored for CPR E 288
- Targeting a Spring Semester Prototype

Problem Statement

- Static Code Analysis is a challenging problem
- Many off-the-shelf solutions
    - Many leave a lot to be desired
    - They are not bespoke for CPR E 288 usage
    - No off-the-shelf “perfect combination” for what the project client needs
- Client needs a code critiquer that can...
    - Be accessed by students and instructors
    - Provide beginner-oriented feedback
    - Ability to give embedded and datasheet-focused feedback
    - Potentially integrate with Code Composer Studio



```
/* XDCtools Header files */
#include <xdc/std.h>
#include <xdc/runtime/System.h>

/* BIOS Header files */
#include <ti/sysbios/BIOS.h>
#include <ti/drivers/Power.h>
#include <ti/drivers/power/PowerCC26XX.h>
#include <ti/drivers/UART.h>
#include <ti/drivers/SPI.h>

/* Board Header files */
#include "Board.h"

#include "DmNodeRadioTask.h"
#include "DmNodeTask.h"


/*
 *  ======== main ========
 */
int main(void)
{
    /* Call board init functions. */
    Board_initGeneral();
```

```
{ \
    Power_init(); \
    if (PIN_init(BoardGpioInitTable) != PIN_SUCCESS) \
        {xdc_runtime_System_abort__E("Error with PIN_init\n"); \
    } \
}
```

Where is it?
I cant find the library for it
OR for others "Macro Expansion"

User Needs

- Feedback
  - Quick answers
  - Confidence in the response
  - Comprehensive
  - Understandable at a basic level
- Security
  - Students can’t cheat results
  - System won’t fail when running
- UI/UX
  - Intuitive/out of the way
  - Quick to use and interpret

Requirements

Functional

- Provide automated feedback based on the antipatterns in the database
- Allow professors to add and remove custom antipatterns to the database
- Allow students to submit code files for critiquing
- Add clarity to existing static C analysis tools
- Use simulated CyBots to analyze the results and function of the code

IOWA STATE UNIVERSITY
Department of Electrical and Computer Engineering

Requirements Cont.

Resources

- Linux server for running the web application
- Server for running the CyBot simulation
- Git repository to hold code and run pipelines

UI/UX

- Clear and intuitive design
- Quick feedback
- Easy to navigate

Engineering Standards

(IEEE)

- **IEEE 1028-2008** - Talks about reviewing code, and our first task of this project is to review the previous team's code.
- **IEEE 2675-2021** - Covers the concepts of reliably, securing, and safely building, packaging, and deploying applications in relation to DevOps. Since our project focuses on having both a frontend and backend, practicing efficient and safe DevOps will be critical to the group's success.
- **IEEE 1016-1998** - Covers the recommended practices for Software Design Descriptors, which are a medium used for conveying the structure of a software system. It will be important to be able to effectively and concisely communicate how the senior design project is structured to the clients and advisor.

Engineering Standards Cont.

(ISO)

- **ISO/IEC 9899:2018** - Explains how C code is made to compile and run. This will be a useful resource to compare code against. Most code that goes against these patterns will include antipatterns
- **ISO/IEC/IEEE 15288:2023** - Describes the terminology, concepts, and frameworks that deal with the lifecycle of a software system. It applies to bespoke and mass-produced systems, so it will be applicable to our project.
- **ISO/IEC TS 17961:2013** - Covers the rules for secure coding in the C language. Since our project heavily deals with identifying errors and antipatterns in C programs, this standard will be an excellent reference for a more memory-safe and cybersecurity standpoint.

Conclusions from User Needs and Requirements Definition

- The software solution involves many types of users
    - Need to ensure that each user group is satisfied
    - The implementation of functionality must be cohesive
- C is a programming language that can easily create security vulnerabilities if not careful
    - Team needs to pay close attention to Engineering Standards to ensure safety, security, and correctness

Any Questions, Suggestions, or Comments?