***Real Time Solutions***

***for Nuclear Fission Reactors***

Proposal, Technical Project

31/03/2017

***Real Time programming***

***SWE 30001***

**Jed Sharman (7657226)**

**7657226@student.swin.edu.au**

Advisor: XXXXX

**TABLE OF CONTENTS**

1. Introduction 1

1.1 Problem Statement 1

1.2 Background 1

1.3 Needs STATEMENT 1

1.4 OBJECTIVE 1

2. Proposed TECHNICAL APPROACH 2

2.1 Requirements 2

2.2 Architecture Design 2

2.3 Implementation DESIGN 2

2.4 Quality Assurance Plan 2

3. Expected Project Results 3

3.1 MEASURES of SUCCESS 3

4. Schedule 4

5. Referenes 5

# Introduction

The Problem:

Nuclear reactors produce tremendous amounts of energy through the nuclear fission of radioactive elements. This process has the potential to fail catastrophically, generally leaving large swathes of uninhabitable while displacing and/or killing droves of people.

To mitigate the potential of such a catastrophe reactors are required to have expansive safety systems designed to reduce the number of reaction to safe levels if possible otherwise to shut down the reactor as safely as possible.

These safety systems require constant and continuous monitoring. Humans are not designed to work under such a load and so an effective computer system is required to maintain the safe operation of these reactors. Due to the gravity of the potential failures combined with the speed at which failures can occur and expand, a real-time solution is required.

Objectives:

The implemented real time solution must,

* Poll sensors at a fast enough rate to detect super critical reactors before they runaway
* React immediately to emergency user inputs
* Accept user inputs for general functionality
* Bring an almost super critical reactor core to a complete shutdown
* Process multiple cores at once
* Mitigate the need for a complete reactor shut down

Benefits:

A real-time solution to nuclear control will have several key benefits.

1. The possibility of reactor meltdowns is majorly reduced
2. The need for complete core shut downs is reduced
3. The number of staff required to safely operate the reactors is reduced with the other flow on benefits of
   1. Fewer people exposed to potential radiation zones
   2. Less room for human error to impact the process
   3. Evacuation congestion majorly reduced
   4. More predictable functionality
4. Amount of necessary space reduced

Just to name a few

## Problem STATEMENT

|  |  |
| --- | --- |
| Element | Description |
| The problem of ... | Nuclear reactors require effective real time solutions to prevent split second catastrophe. |
| Affects ... | Land owners, plant investors, plant workers, residents and visitors of the local region. |
| And results in ... | Reactor failures can cause injury and death to workers, residence and visitors. Drop land prices and render local areas uninhabitable. Call for hefty fines and compensation suits against investors and workers.  A general reduction in quality of life for all involved. |
| Benefits of a solution ... | The solution reduces human error potential while increasing productivity and helping to mitigate the chances of severe failures. |

## Background

Nuclear power has a long history dating all the way back to the 1930’s where it was theorized that the exceptional amount of energy released by a nuclear fission could be used to power everything from generators to automobiles.

The very first use found for the newly found power source was as an instrument of war. In 1945 the United States of America dropped two explosive devises named fat man and little boy on Japanese cities. The primary payload of the two devices were nuclear fission based, the effectiveness of the bombs was beyond the expectations of both the developers and the public who witnessed hundreds of thousands of civilians being annihilated without warning. As a direct result of this first introduction to the nuclear age both public and private concerns have indevoured to keep the restriction and scrutiny of all nuclear activities harsh.

In 1979 a combination of secondary system failures and human error (an emergency release valve was left open) resulted in a nuclear disaster at the three-mile island plant in Pennsylvania, USA. The accident was primarily caused by poorly design user interfaces indicating to plant workers that there was a need to vent coolant which led to a pilot operated valve being manually opened and spewing radioactive coolant into the environment.

In 1986 a poorly planned and executed stress test on a reactor at the Chernobyl nuclear plant is soviet Ukraine resulted in the worst nuclear disaster in history. With several of the safety system disabled, human error resulted in a flash point steam explosion which ruptured the reactor and scattered nuclear material across several countries.

In 2011 the Fukushima Daiichi nuclear plant in northern japan was subject to three reactor meltdowns a after combination of natural disasters. An earthquake shook the plant on march 11 to which the reactors initiated an automated self-shutdown. Following the earthquake, a tsunami hit the plant knocking out its secondary generator bank, this caused a failure in the cooling systems which resulted in the meltdown of reactors 1-3.

In light of the numerous disasters surrounding nuclear facilities and their control systems the scrutiny to which any associated system is held is intensive. This impacts the project and above all else the project must be completely bug free upon release.

Though accidents such as three-mile island and Fukushima were caused primarily by poorly designed and maintained control systems these outdated systems remain in use today. With the costs associated with decontaminating and upgrading plant systems most energy providers choose to simply try and patch up existing systems.

While modern systems are designed with a digital mindset the majority of older system remain analogue making them less robust and considerably less reliable than their digital counter parts which still require a large amount of human interaction (e.g. three-mile island) and allow for human decisions to supersede safety protocols (e.g. Chernobyl, three-mile island).

Problems Identified:

* Systems outdated
* Operate in an analogue time frame
* To costly to reconstruct entire infrastructure
* Systems ambiguous and hard to interface with
* Systems superseded by unqualified personnel

## Needs STATEMENT

The current organization and implementations are out dated and non-standardized, they rely on software, hardware and plant which in its prime gave way to catastrophic disasters and has since aged several decades.

The planned project needs to:

1. Incorporate real time digital components
2. Be able to maintain schedules including several independent reactors from a single terminal
3. Be adaptable to being implemented and interfaced with both outdated and modern infrastructure
4. Have functionality which includes dominant safety protocols
5. Includes functionality restrictions incorporating, some kind of soft lock.
6. Has an interface which is completely unambiguous.

## OBJECTIVE

Have a release ready program written in the ada programming language. The program should include all of the following functionality:

* User output
  + Ability to monitor current system state
    - Readouts of system variables
    - Readouts of system trends
  + Ability monitor current system behavior
    - Readouts of how the system is reacting to its current state
    - Readouts of why the system is reacting to its current state
* User input
  + Ability to give the program commands
    - Inputs are interpreted as commands which call on system functionality
  + Ability update system behavior
    - Inputs can force the system to act against its better judgment
    - Inputs can force the system to act in a pseudo “manual” mode
    - Inputs can not deactivate safety protocol warnings
* Password restrictions on input
  + Ability to lock functionality to certain privilege levels
    - Entering a dialogue environment requires inputting a password
    - Inputs are rejected if they attempt to access functionality tied to a higher privilege level than the current
  + Ability to override commands from higher privilege levels
    - If a functionality is accessed with a locking flag that functionality is locked to that privilege level or lower.
* Continuous monitoring of current state
  + System updates independently of user input or output
    - System polls sensors in independent threads
    - System uses soft interrupts for sporadic sensors
  + Monitoring is safe real time
    - System should monitor with minimal delay between when a reading is available to be read and when it is available to the system
    - System should maintain data integrity and not allow concurrent read/write operations
  + Monitoring has a high enough resolution
    - System should have minimal delay between sensors readable values changing and sensor being read
  + System logic should be applied in real time
    - When the system state is change, system logic should be applied and reaction calculated
    - System reaction should be updated as soon as possible with exception to maintaining data integrity
    - System should have the ability to override reaction if necessary
* Continuous system output
  + System outputs should implement system reactions in real time
    - System should call functionality as specified by the current reaction
    - System should implement new reactions as soon as the current reaction is updated
  + System outputs should act independently
    - System outputs should run on separate threads using soft interrupts to be initiated
  + Zero down time
    - The program should be able to be run concurrently with another version of itself.
    - The program should hand shake and communicate with concurrently versions of itself
    - The program should, on user input, relinquish control of the system to another concurrently running version of itself after a handshake has been established

# Proposed TECHNICAL APPROACH

For the development of the project a testing bench will be required, this includes a simulated environment such as the program might encounter in lifetime operations. For this an independent process, will need to be developed incorporating objects capable of updating sporadically

The project will be built using the following steps:

1. Project test bench development
   1. Includes simple objects which update their own state independent of other processes, therefore these objects should be run in their own threads with atomic input/output
   2. These should be a representation of a fully functioning reactor bank
      1. The reaction material
      2. The control rods
      3. The reactor housing – contains the reaction material and the control rods. Tracks the flow and distribution of salt around the reactor and works to update the coolant tank
      4. The turbine
      5. The coolant housing – houses the turbine and the coolant. Tracks the flow of coolant and works to update the turbine.
      6. Actuators – these are objects which can change the behavior of the reactor
         1. Reaction control - percentage of reaction material able to react
         2. Pressure release – opens release valve, venting coolant
         3. Reactor dumb - drops reaction material into a sealed vat of control fluid and seals it. \*\*\*Last Resort\*\*\*
      7. Sensors – these are objects which can read the current values of different objects
         1. Salt temperature levels – read from the reactor housing
         2. Salt flow rate – rate the salt is moving around the reactor – read from reactor housing
            1. Rate reservoir to reaction chamber
            2. Rate reaction chamber to cooling tank
            3. Rate cooling tank to reservoir
         3. Coolant temperature – read from coolant housing
         4. Coolant pressure – read from coolant housing
         5. Coolant flow rate – rate the coolant is moving around the reactor – read from coolant housing
            1. Rate reservoir to cooling tank
            2. Rate cooling tank to turbine
            3. Rate turbine to exhaust
         6. Turbine speed – read from turbine
   3. Should include input and output streams which can be used to read/write to the sensors/actuators.
2. Project main body development
   1. Hierarchical objects should be used to implement the programs functionality
   2. User interface – object
      1. Requires password to access functionality (initiates with minimum privileges)
      2. Reads user inputs
      3. Interprets inputs, if valid command attempt to access command with current privilege level
      4. Displays system outputs to the user
      5. Displays system state to the user
   3. Command – object
      1. Has unique identifier
      2. Has priority
      3. Has list of applicable flags
      4. Has current privilege restriction (int)
      5. Calls other objects and their methods to achieve functionality
      6. Checks the privilege level the command is being called with
      7. When called runs in its own thread
   4. Command List
      1. Has list of commands
      2. Is a globally accessible environment object
      3. Holds the persistent copy of each command
      4. Is created upon program initialization
   5. Agent
      1. Implements reactionary logic for users
      2. Has a unique identifier
      3. Stores several variables which can be manipulated by commands
         1. Bool, \_shutDown
         2. Double, \_safeReactorTemp
         3. Double, \_reactorTempSafetyMargin
         4. Double, \_safeCoolantTemp
         5. Double, \_coolantTempSafetyMargin

Etc…

* + 1. Can maintain one reactor
    2. Has internal model of system state
    3. Continually updates internal system model in real time

*[Define clearly a well thought-out and solid technical plan for applying information technology to the proposed project. This section should include a description of the methodology to be used to complete the project, a specific plan for gathering requirements, an architecture design, best practice for implementation, and quality assurance.]*

## Requirements

*[Present the requirements as understood at this time through contacts with the stakeholder. Include a high-level diagram such as a use case system diagram or block diagram to capture the situation being addressed.]*

## Architecture Design

*[Explain the technology to be used in the project. Describe hardware, software, or network components as relevant and as understood at this time. Draw a high-level architecture diagram to illustrate the proposed system components and the relationships between them. ]*

*[OPTIONAL CONTENT: Include an as-is system or block diagram.]*

## Implementation DESIGN

*[Describe your methodology for implementation.]*

*[OPTIONAL CONTENT: Include a best-practice approach to be followed.]*

## Quality Assurance Plan

*[Describe the potential risks related to the software quality. Provide the project management plan to enable quality. Describe the salient, planned testing considerations.]*

*[OPTIONAL CONTENT: Include any analysis plan for usability and acceptance testing.]*

# Expected Project Results

[List deliverables expected to be produced for the project, for the CIS Department, and for the stakeholder.]

[OPTIONAL CONTENT: Identification of any currently operational aspects of the project and how the behavior would change.]

*[OPTIONAL CONTENT: Itemize expected, measurable results of the effort.]*

## MEASURES of SUCCESS

*[Describe an assessment plan to identify the degree of achievement obtained by the operation of the project. List a measure, its relevancy to the organization, its current value, and its projected improved value. A measure should also correspond to the stated objectives of the project.]*

# Schedule

*[Attach schedule, a Gantt chart or time-table would be preferred.]*

# Referenes

*[include your references]*