This research is aimed at developing a new tool to facilitate efficient development of control software for safety-critical applications. Like Computer-Aided Design (CAD), Model-based development (MBD) tools offer a graphical programming environment where the user develops the program as a graphical model from which the code is generated automatically. Our research focuses on Simulink, a tool by The MathWorks. The C code is generated automatically using another tool called Real-time Workshop.

**Problem Statement**

Developing tools and technology for producing safety-critical software.

This research is aimed at developing a new tool to facilitate efficient development of control software for safety-critical applications. Like Computer-Aided Design (CAD), Model-based development (MBD) tools offer a graphical programming environment where the user develops the program as a graphical model from which the code is generated automatically. Our research focuses on Simulink, a tool by The MathWorks. The C code is generated automatically using another tool called Real-time Workshop.

**Softwate Reliability**

**Expansion of Capabilities**

- F-4 (1960): 8% of capabilities provided by software
- F-22 (2000): 85% of capabilities provided by software

**Growing Size of Software in Industry**

<table>
<thead>
<tr>
<th>Year</th>
<th>Code Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>1,000</td>
</tr>
<tr>
<td>1980</td>
<td>10,000</td>
</tr>
<tr>
<td>1990</td>
<td>100,000</td>
</tr>
<tr>
<td>2000</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

**Importance of Software Verification**

Mars Polar Lander – Powered down 100ft above the Martian surface – most likely a software bug.

Ariane 5 Disaster

Complexity of software grows – reliability an important issue. Infamous Ariane 5 disaster, arguably one of the most expensive software bugs in history.

**Non-Functional Requirements**

- Usable on any system running the Eclipse Platform
- Process large model with over 100 blocks in under a minute

**Test Results**

**White Box Testing**

- Testing of individual methods and classes
- Uses JUnit framework

**System-Level Testing**

- Testing of system as a whole
- Input is a single set of files
- Intermediate output from each stage can be saved
- The source of a bug can be tracked to the component

**Test Results**

- Revealed a number of bugs in the code
- Identified blocks which are not supported by our code
- Test succeeds in all models using only supported blocks

**Conclusion**

Our project demonstrates that it is possible to automatically test generated code for errors. It is capable of matching nearly all types of blocks. It could be developed into a robust commercial solution to save test engineers a great deal of time when auditing automatically generated code.

**Project Plan**

**Work Breakdown**

1. Create Simulink Models and C Code
2. Manually Identify Simulink Blocks in C Code
3. Write Code to Obtain an Abstract Syntax Tree
4. Write Code to Detect Blocks
5. Write Code to Output to DOT file
6. Verify Generated Model is Functionally Equivalent
7. User Interface
8. Testing and Debugging

**Deliverables**

- Requirements Specification
- Engineering Specification
- Source Code
- Project Poster
- Project Plan
- Design Documents
- User Manual
- Website

**Faculty Advisor**

Dr. Sunaj Kathari

**Team Members**

Daniel De Graaf
Corey Kleinmetsel
Korte Koenbundj
Benjamin Miller