**Chapter 1 Introduction**

* 1. **Introduction**
  2. **Definition of the problem –Review of features needed for SCD**
  3. **Hypothesis**
  4. **Project Aims**
     1. Aim
     2. Objectives
  5. **Plan**
  6. **Summary**

**Chapter 2 Background research**

**2.1 Introduction**

**2.2 Problems with Development Methods**

2.2.1 Agile

2.2.2 Waterfall

**2.3 Programming Languages used for safety critical systems**

2.3.1 Features

2.3.2 Usage in Industry

2.4 **Railway Crossing**

2.4.1 How they work

2.4.2 safety Features

**2.4 Summary**

**Chapter 3 - System Design**

**3.1 Requirements**

**3.2 System Overview**

3.2.1 Description of system

3.2.2 UML Diagram

**3.3 Design of Animation**

3.3.1 Sketch of Design

3.3.2 Failures in Animation

3.3.3 Success in Animation

3.3.4 Communication between Java and Ada

**3.4 User Interface Design**

**5 Implementation**

5.1 Implementation of Railway Crossing

5.1.1 Basic Structures of types in ada

5.1.2 Modelling Failures

5.1.3 Modelling Success

**5.2 Animation**

5.2.1 Design Aim

5.2.2 Design Implementation

**6 Testing**

6.1 Railway Implementation Unit Test.

6.1.1 Testing Non Proofed parts

6.1.2 Testing animation

**6.2 Spark Tools**

**6.3 Proofs by spark.**

**7 Evaluation and Results**

# Introduction

A Safety-Critical System is a System in which a failure can result in Losses of millions, it can cost lives and injure people. Such Systems require very careful planning when developing them, some clients of programming languages may stick with such languages as C in developing embedded software.

However, languages such as C are very hard to prove correctness of a C program. Such as accessing a pointer, or trying to allocate Heap Space. What if there is no memory? What should the system do? Such faults as even divide by zero errors can lead to the compiler errors, Overflow Errors Such as *the Boeing 787 Dreamliner*[1] <https://www.engadget.com/2015/05/01/boeing-787-dreamliner-software-bug/>

<https://s3.amazonaws.com/public-inspection.federalregister.gov/2015-10066.pdf>

In which an Integer Overflow bug will eventually occur within “*248 days of continuous power”* AOL (2017). This means that it resulted in a loss of all AC electrical power.

These Types of errors are 100% not allowed to happen in Safety-Critical Systems.

Such problems may only be solvable at runtime and which excessive testing, trying to throw a range of values that the subprogram may take to find these problems.

However, there are some Languages that can be used to Tackle these problems before they arise. One example is Spark which is a subset of Ada, Checked C, Spec Sharp, Safety Critical Java. These languages offer Tools for proofing that a program is free of errors, by generating proofs to ensure no matter what happens that the programs are never going to be executed by a divide by zero for example or an overflow.

# Definition of the problem

In this paper, I will be using Spark to develop two case study’s. One being a light bulb example to get an easier of idea of what pre-conditions post-conditions and other features of spark and then develop a bigger study of a rail way crossing simulation. These will then be compared to Java in which I have 2+ years’ experience in using and comparing them to other similar languages such as Spec Sharp in background research.

I want to use these findings to then find what types of features are needed to develop safety critical Systems, and current tools for Java that can be used to develop a safety critical system and features that cannot be used. I want to also compare the usability of spark and java and Learnability.

This could be used to identify features of Spark and Spec sharp to be used in future development of Safety-Critical Systems and other types of systems which might not be Safety Critical such as a bank System. This can then be considered when clients want to choose a programming language for their system.

# Project Aims

**Aim:** Investigation of approaches to safety-critical systems development.

The Aim of the project is an Investigation of approaches in which I look at a range of software development methods, draw comparisons to them in which I will choose the one, which is most beneficial to safety. I will then develop a very small case study, and figure out what features are needed for safety critical development and compare Spark to Java in terms of usability, ease of use and errors.

**Objectives:**

1. **Research into software process methods for safety-critical Systems.**

Researching into the correct software process method to use will be critical to the project for making sure areas are covered, at least four should be researched before starting anything else to make a comparison between which is best for a safety critical System.

1. **Research into languages that are used for Safety Critical Development and see what features they offer.**

Researching current languages used for safety-Critical development will involve looking at what kinds of languages are out there to date. I want to focus on the languages that companies are more likely to use to develop the systems I will do this by trying to find sources, which point to some popular languages.

1. **Develop a full evaluation Criteria which evaluates both programming languages for a comparison**

This will be based on the research I have gathered together to find a full evaluation Criteria which should be based on things such as features, code coverage, usability and correctness.

1. **Learn Language which is popular for Safety-Critical development**

I want to learn one Language that is appropriate for safety-critical development, in this case will be spark and knowledge of Spec Sharp.

1. **See what features are needed for Safety-Critical systems**

I want to be able to identify what features are needed to fully proof that a language is free of runtime errors by examining what features are most important to do this.

# Background Research

# 2.1 Introduction

In order to fully compare these languages a lot of background research had to be done to improve my knowledge of them. A lot of my learning from languages such as Spec sharp, checked C and Safety Critical Java are done here. To improve my knowledge of such languages and most important for demonstration purposes I have used full explanation of books for Spark [Spark Ref here] and for the others have used specifications which is the only resources I could find on these other languages.

I want to use this research to make comparisons on the Languages. See what features each have to offer and if these are meet safety Critical requirements.

I can also make some assumptions on usability by comparing the code to Java in which my opinion is easier language to generate good code, and easier to write in.

# 2.2 Development Methods

Development methods used in industry can affect the overall quality of a project, where agile is for very fast moving industry waterfall is more beneficial towards projects which can afford some slack time, formal methods can be used to compare spec directly to code by actually proving the code meets the specification.

Formal development is actually used often for Safety-Critical development, saying that the program is fully proved to be identical to the specification may sounds good but if the specification is worded incorrectly behaviour may be implemented which was not previously wanted. I will talk about these in this section and what the advantages are and disadvantages of using them are using background material to support my discussion.

# Agile

Agile methodology is used a lot in the industry, nearly every company which needs to adapt to new technology needs to use it. Agile development introduces that a company which needs to cope with ever changing requirements and technology requirements means that code may have to be released with patches, which is not acceptable in safety-critical development.

A popular book Somerville which is used a lot by software engineers and recommended reading at Newcastle University states that because of rapid development “*businesses are willing to trade off software quality and compromise on requirement’s to achieve faster deployment*”. [Summerville Ref here Page 57]

Summerville also sees my concerns with Agile used as process for Safety Critical Development in that by the time the software has been released it may have changed. “*By the time software is available for use original reason for its procurement may have changed so radically that the software is effectively useless*” [Summerville Ref here Page 57]. This result of radically changing requirements result in money wasted, and if the purpose changes the whole system changes effectively meaning a new project would have to be started.

Even if Agile is used Money will always be wasted by advances in technology. If a company which develops Safety-Critical Systems have to keep making other iterations it’s hard to keep up with this. If one system is developed it would be unacceptable for then requirements to change and then other technology be bought to cope with this change.

Agile is also based on iterative development which then is improved based on feedback [https://pdfs.semanticscholar.org/69b1/9ddc8a578f4c63d1dfe15252a465ee12fe5d.pdf].

This is not needed for Safety-Critical development it may be needed a little bit but nowhere near as much as agile. If needed it can be incorporated into the project of a Plan-Driven Development.

Based on these and my own thoughts, Agile should not be used to develop Safety-Critical systems. Based on:

* Too much changing of requirements, Safety-Critical system requirements should already be known before development starts.
* Agile development can often result in bad code, which cannot be the case where people’s lives are at risk.
* Too much dependence on customer involvement.
* Patches should not be part of the process
* Is not a plan driven model.

# 2.2.2 Waterfall

Waterfall is a planned type of development process, this means that all steps before starting a project are planned and that changes are not to be made during actual lifecycle of the process.

Within the development of software using this model it is necessary to be able to finish each part of the project before starting the next unit of work [Summerville Page 31].

Because the waterfall development model is so detailed before the actual project is undertaken it means that development is very secure, this is all of course assuming as with all projects that, Requirements, work distribution and time management is correctly measured.

Each unit is treated in sequence and the next sequence should not be started until the previous section is completed [Summerville Page 31]. It ensures that no other process should be started unless it has been completed ensuring each stage of the model is completed with full confidence that it is correct.

However, because of previous changes in the model it means that certain parts of development may have to be reworked because of changes [Summerville Page 31]. This could cause problems and misunderstandings because of the constant reworking of documentation, which will lead to some parts being frozen for others to start being worked on.

This can cause slower development later as documents get refined, however has the advantage of pausing uncertain changes before they can be undertaken. Afterwards these things can be reviewed and updated.

However, because of freezing it can lead to software which is essentially wrapped around to fix problems adding redundancy which is not previously needed.

Overall, I think that waterfall could be used within Safety Critical Development Because of:

* Plan driven process model
* Every stage should be completed before next task is started
* Each unit is done in separate stages
* Every Unit is signed off before next stage begins
* Updates freeze other parts of development to be fixed later.

# 2.3 Programming Languages used for safety critical systems

Programming languages can vary in Style, storage of Types and data manipulation, however different languages can affect behaviour of systems. For example pointer manipulation is not allowed in languages such as Ada and Spark. This kind of difference should be noted before starting development, as it can leads to problems later on.

For example if your program is written in full Spark, you can have confidence that your program will be able to be free of errors, regarding that the parts of the program you have written are not separate Ada packages with Spark disabled.

There are many languages that can be used for Safety Critical development, technique speaking you could use any language you wanted to, however certain features of languages make stronger choices. In terms of features the language naturally provide and some features can be used to ensure that your program will not fail.

Most software companies would rely of software checks to be put in the code, this can be beneficial and still are needed regardless of what language you would choose. However, some checks are used due to poorly written code and when Safety-Critical systems are developed it would be ideal to write code that you are sure will work and then worry about adding redundancy to the system such as a voter, or system checks.

Such languages can provide the answer to reason as to how your program can be made to be error free, such as Checked C, Safety-Critical Java, Spark and Spec Sharp. All these languages have something in common, they strip away features that may be accessible in full set of the languages and hold a strict subset of the languages. This means that some languages may not be able to make use of pointer manipulation, but this is all working towards a safer program. They provide pre and post conditions which restrict the entry point and the exit point of a program to specified values. This will be evaluated here. I will find common methodology between them which will help me decide what is it that is needed to help develop a program which is free of runtime errors.

# 2.3.1 Features

Checked C

Although Checked C has not been realised yet there are still points to talk about with this new form of C. C is one of the most widely used languages for embedded systems. Although there is not much information on the web for checked C there is a site on Microsoft site which contains some details for checked C.

It explains how such errors such as buffer overruns and incorrect type casts, that can be made in C. [https://www.microsoft.com/en-us/research/project/checked-c/] This is big as you may not know that this is possible even through testing if proper testing is not done, it could be right up until two months into someone using the software for errors to be discovered.

The Aim of Checked C is to be able to detect these errors before even running or while in execution by letting clients add checks to the program.

They also address how common it is for programmers to make mistakes using pointers, saying that a program reads or writes the wrong data[https://www.microsoft.com/en-us/research/project/checked-c/] Which could send the program into an erroneous unrecoverable state for most software languages such as databases where a program could write data incorrectly and corrupt the storage for that data which could be an important bit of information. In Safety, Critical Systems it would depend on how important that information is, if it is important than it could lead to failure, for example a railway when encountering a problem would stop would just stop the train, which is the last resort, but a problem such as overflow may cause the program to crash and leave train driver to do all the work.

They make it better by allowing the client to better describe the how they want to use the pointers and the range of memory occupied by data that pointer points too [https://www.microsoft.com/en-us/research/project/checked-c/]. This gives the programmer full power to access blocks of memory safely instead of trying their best and hoping that the data block is safe to access, and describe what kind of information is likely to be flowing at that point, like the assert pragma in spark is allows the user to describe some information and adding checks at that point. In checked C it is then checked at runtime to pick up mistakes caused by the user, if there is a problem it is then addressed such as accessing wrong data, Chceked C would then pick this up as the Client did not want to do this it. It would know this as the user has described what kind of data is expected and where and checked C would tell the user this is incorrect, instead of silently acting on a dirty read.

They also add bounds checking, for example when you specify how big a value is in Eclipse and that value goes above a certain amount it will give you and error before being ran. This is all helping towards a safe Language adding towards static checking.

In the document, it also compares itself to Java and C# explaining how they already have bounds checking which I have already shown, But it pulls itself away from Java saying that these are automatically add information and says that Checked C will allow the programmer complete control over the needed bounds-checking and how information flows through the program[https://www.microsoft.com/en-us/research/project/checked-c/]. This suggests the use of pre,post and depends contracts just like in Spark although not 100% sure up until release it sounds a lot like it due to explanation of flowing through the program, which is what depends does in spark, but also adding bounds checking such as pre conditions do in Spark.

Spec#

Spec# is another version of checked C, but Spec# has not had a major update since 2011 which has fallen behind the latest version of Visual Studio, so now people must install old version of it to get it to work and no real documentation or guides on how to install it. It is mostly learnt on Microsoft’s website where someone has written an introduction into learning spec Sharp.

Spec# builds on how specifications can be enforced dynamically and statically [Spec Sharp Ref Page 2]. This is very much like Spark, as Ada is a spec based language which allows programs to be written in a way so it complies with real life specification and requirements. Also, it suggests static verification which means that a program can be tested before runtime. It also says it checks that subprograms compile to their specification.

It states that it performs modular verification, which means that it can be applied to pieces of a program separately [Spec Sharp Ref Page 2]. What it relies on is the fact of state changes in the system, for example an insertion in linked list’s. You would describe how you’d insert the data into the list and changes made by representation that in the state of the data structure eg updating the value and moving the iterator.

It also states how much easier it is to program from the specification, verifying the subprograms in accordance to the spec, given that it is written correctly helps C sharp verify the programs correctness. Below is a list of features in C# All from [Spec Sharp Ref Page 2] and [http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.933.8240&rep=rep1&type=pdf]. I will provide a basic description in what I believe are some of the important aspects to notes in Spec Sharp.

Spec Sharp Feature Table Figure 1

|  |  |
| --- | --- |
| Feature | Description |
| Non-Null Types | Uses non-null types and possibly non-null types to distinguish between them, by using the type followed by a question mark eg string?. Includes the String object plus the null.  It gives the user a flag message saying that there would be an error when trying to reference an object would could be null when you try to do operations on it. The Type checker for Spec Sharp does this for you, and the only way to fix it is to ensure that the string is initialized to some value other than an null.  Or could possibly type cast it to a That type to ensure that string operations can be performed on it, a check is therefore in place to verify this and it will due to the type being the operate type. |
| Method Contracts | Contracts are very common in these types of languages.  These specify what kinds of bounds are allowed in an entry point of a subprogram and the expect bounds of outputs, just like pre’s and posts. These are written in the specs of a spec sharp program. Contract means that if the correct precondition is supplied it is guaranteed that the subprogram will pass due to the precondition correct.  It is worth noting that in the spec you can use the keyword result which would give you reference to the result of the method, which is powerful when adding checks to help the compiler check that it is correct. The is also construct old which refers to the state before the entry of the method.  Tools used are then checked to see If they match the correct type and be of appropriate type which would be Boolean for preconditions and post conditions. |
| Inline Assertions | Assertions give a program a way to describe their code, it is very good at this. Where code such as assertions in Java used to be used in comments to enforce some knowledge of a given program, now it can be used in testing or runtime of a program.  For Spec Sharp, it can be used more than that. With spec, sharp a condition can be statically checked by the prover. This is a good feature. Being able to prove a statement about a given program without having to give that statement a range of values is so much easier to be verified rather than using unit testing. You may miss things out with testing where the verifier ensures that no statements are missing. Spec sharp would issue a complaint when an assertion does not hold. |
| Loop Invariants | Loop Invariants describe a set of states that the program may be in at the beginning of the loop iteration.  It specifies some condition that will be true each time on entry to the loop and on each iteration. The condition must be sufficient to rule out any invalid states, the spec sharp prove will complain otherwise. The point of check is that it holds on entry and on every new iteration. The verifier will explore all ways through the loop body to determine what may be assumed to hold after the loop.  The quantifiers such as forall and exists are common for these types of data collections, where in a loop you have one or more executions it is good to make use of forall and exists quantifiers when proving the loop invariant to the post condition. Important thing to note Spec Sharp does not check for Overflow. |
| Object Invariants | Object invariants can limit what kind of values an object may hold. For example, may be a case where objects may take values from 0 to 10. could be used in cases where Invariants need to hold on 1 or more objects at the same time. Eg when a value is less than 0 then the other object must contain a certain value.  The Spec Sharp verifier will see this and make sure they’re always in line and validate checks. Expose blocks are used to ensure that an invariant can be broken within a block without the compiler arguing that the invariant is broken at that point, they then become mutable objects which are not subject to invariant checking. |
| Arrays Non-Null | Arrays require special treatment to ensure that the elements you are putting into that array are of the correct type.  To allow spec sharp to correctly verify arrays a proper pre-condition needs to be satisfied that only arrays of a certain type are to be accepted.  Non Null elements means that an array could be of the type Person[] but the actual elements could be null instances. The Spec Sharp prover follows the new allocation, the NonNullType.AssertInitialized is used to indicate that the program has completed initialization. Until then the checker will not give the array the declared non-null type until then. It ensures that the Array is initialized properly, in that all elements contain some form of data of that type.  The NonNullType.AssertInitalized can be used to assert that the array is of NonNullTypes and then the program verifier can verifier this. It is helpful when the initialization may be done in a sequence instead of at declaration.  Ensuring that Arrays of Non-Null elements are not accessed later. |
| Static Verification | Static verification is used to enforce such things as non-null types, this will enforce run time checks for contracts that are generated by using tools such as pre and post conditions, and then record the contracts as metadata.  The SscBoogie is a static program verifier used by Spec#. This generates verification conditions from a spec# program. It then uses a reasoning engine to analyse the verification conditions generated by SccBoogie to prove things about the program and trying to find errors in the program. [http://www.rosemarymonahan.com/specsharp/papers/SBMF2010.pdf]  Adding relevant invariants, contracts ect help the tools prove things about the program, it’s a chain of tools which all come together to prove parts of the program.  If any violations are picked up, they are then put in a list of verification error messages for the programmer to fix.  These can be tested again until completion.  The Boogie program goes through several transformations, one example from [http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.933.8240&rep=rep1&type=pdf] is cutting all loops to derive an acyclic control flow graph. Ensuring a graph which has no cycles, it uses havoc statements causes theorem prover to consider arbitrary loop iteration as it is important for a loop to be making progress, so it is easier to discharge the verification.  These static verifications are very powerful, if the programs reasoning is correct and not taking component failures into account, then a programmer can have confidence that the program is logically correct and the chances of an error occurring are reduced. |
| Run Time Checking | Because of the tools used for static Verification, it can take a massive performance hit when trying to compile, in Real time systems it could cause trouble.  Spec# takes care of this by turning pre and post conditions into inlined code, which is done for performance and avoid extra methods and fields in compiled code.  This is then used to separate the contracts and code that comes from rest of the spec# program, as some parts of the program may not have contracts.  The inlined code evaluates the conditions and if they’re violated it will throw a contract exception. |

Java

Safety-Critical Java

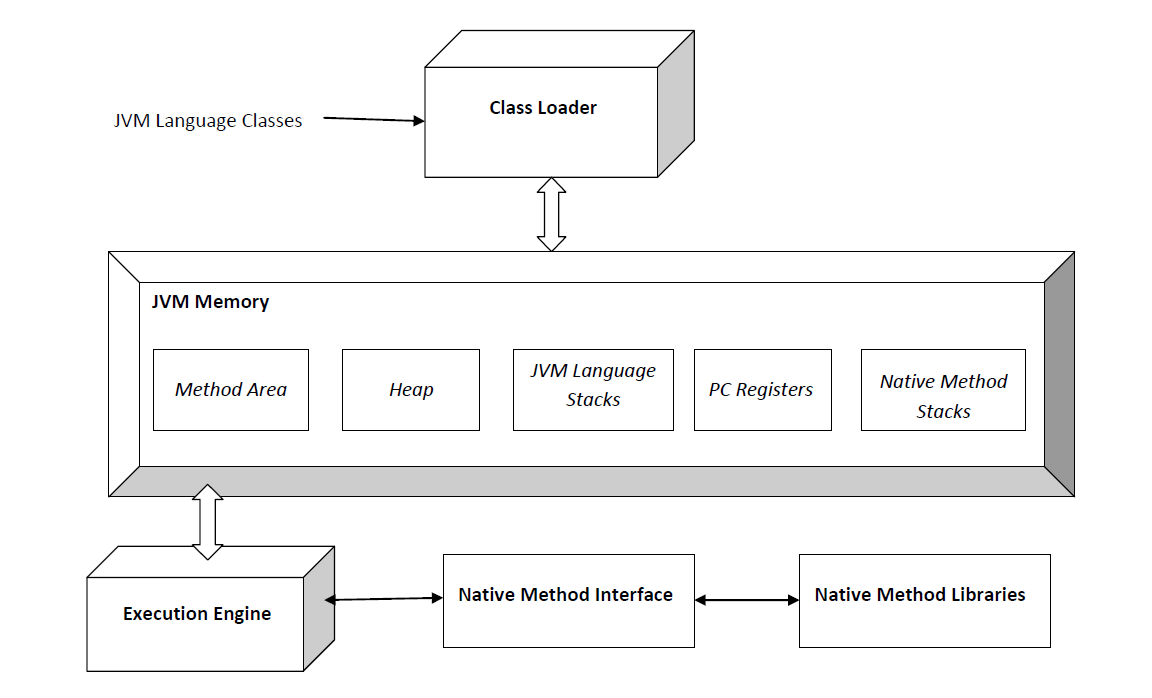
Safety-Critical Java Is not yet released to the public it is still in the planning stages.

Java is one of the most commonly used programming languages up to now. On tiobe it is currently the most used Language to date [ <http://www.tiobe.com/tiobe-index/>]. From previous use of Java, it is easy to say now having experience from other languages it is a very easy language to learn and write in. With a lot of Plug in’s such as PMD and CodeBugs to enable static analysis of code which I will talk about later.

The reason as to why this has been developed is to address the needs for safety-Critical Software.

Safety-Critical Java specification states that to be successful that it needs a smaller and highly predictable set of Java Virtual Machines Also states it must exhibit freedom from any exceptions that cannot be handled [http://download.oracle.com/otn-pub/jcp/safety\_critical-0\_94-edr2-spec/scj-EDR2.pdf?AuthParam=1489342923\_ee7fd1e36e78cfa2d98455e6552dc194].

Figure 2 [https://en.wikipedia.org/wiki/Java\_virtual\_machine#/media/File:JvmSpec7.png]



The Java Virtual Machine Pictured Left, enables Java to run across multiple platforms if the Java runtime environment is installed on the machine. It is an abstract Specification, it also means that Memory cannot be manipulated as in C. The JVM manages Memory layout for you. While this is good, there may be times where you would like to edit Objects in memory as with C such as using the free method in C.

They have already managed to construct Systems using standard Java Technolgy, which is a good step into proving that Java could be used in Safety-Critical Technolgy. Level D and Level E systems have been created which explains that, In these systems behaviour which is not expected by the system would cause minor Defects in the program, eg it would not entirely break the systems.

This may seem minor but correct or detecting systems which are in an erroneous state is important to be able to reduce the error and try correct it. However the aims are to achieve systems with higher capablities.

With Safety-Critical Java they want to enable to usage of Java in these types of applications but not mission critical Applications in which a failure of a one part would lead in a mission wide failure.

Looking at the Programming in SCJ report One problem of Java is the way it lays out memory. I said before that it would be important to have some control over memory management in Java, but in this document it says that because of the way Java Layouts it’s memory and also the finalization through garbage collector the impact of a garbage collector may have a negative impact on anything anaylizing the timing properties of the system [https://www.cs.york.ac.uk/circus/hijac/code/ProgrammingInSCJ.pdf].

It introduces that RTSJ allocates a new type of memory Scoped Memory. In which objects with well defined life span are created [https://www.cs.york.ac.uk/circus/hijac/code/ProgrammingInSCJ.pdf]. It solves the problem by allocating outside the heap which means that it is not subject to garbage collection. These are the sorts of methods taken in Safety-Critical Java.

PMD

PMD like Spark will staticlly anaylise your source code. It is very easy to install via a plug-in on intellj. There a number of ways to use PMD, it will anaylise all your source code looking for bugs. It is also compatable with the lastest version of Java.

PMD does performs syntatic checks on program source code to find errors in the program’s, PMD’s let down is that it does not try draw a dataflow graph[http://www.cs.umd.edu/~jfoster/papers/issre04.pdf]. A dataflow component attempts to draw how data is being flowed through the program, It can then find certain areas where an object depends on another object and in turn which variables get initalized in a if statement. Certain branches may not be initalized but that unitalized variable may be used later on, which can cause problems. This is more solveable using a dataflow component which PMD does not offer.

It will try and find possible Bugs in your program, for example maybe you have written a try catch but have not implemented it. PMD will pick this up and ask you to implement it, having this as a static anaylizer is very handy espcially since running PMD will flag the errors, as opposed to just underlining them in a different color as PMD and eclipse does. [ <https://pmd.github.io/pmd-5.5.4/index.html>].

It also catches Declaration redudancy it will complain if some parameters never change in the lifecycle of the program.

PMD will pick up on Dead Code, Eclipse does this too but does not force you to remove it, dead Code wastes resources and may be taking up space even though a variable or method will never be executed, in Systems such as safety-Crtical System resources may be limited and Dead Code is not ideal [ <https://pmd.github.io/pmd-5.5.4/index.html>].

Suboptimal code, this is performance reasoning. For example adding onto a string, which creates a new String in memory each time. That is very wasteful PMD would complain about this and maybe to use a Append to add more information to a string.

Overcomplicated Expressions, This helps remove some complicated expressions for example Writing a method that returns a value but actually returns the same value every time, tools would give some suggestions such as should be made void. Also tells use when certain methods can be made private to give them the right access. [ <https://pmd.github.io/pmd-5.5.4/index.html>] For an example of expressions I have some code here the actually generate a conrol flow issues. IsManual == false is a boolean expresstion to check if a car isManual, the tools pick this up and complain that this should be simplifed to !isManual to simplify the expresstion.

Also looks for Duplicated Code, Copying a pasting code is generally note a great idea, once you start copying and pasting code it means that code is not written to good quality. PMD will ask you to correct this of analysing the code. [ <https://pmd.github.io/pmd-5.5.4/index.html>]

PMD does all this without running the code, and if you agree with PMD it will change the code for you without you having to write a single correction.

However with PMD there can be some anaysis which actually reports that correct code is wrong, which is okay to ignore but shows that PMD can produce incorrect results. It could also miss a few common errors in code for example in this document PMD does not pick up that on comparison of strings using the == sign in Java which Is wrong, but Findbugs does pick this up [http://www.cs.umd.edu/~jfoster/papers/issre04.pdf]. This will be investigated futhure in later studies of using PMD and findBugs.

FindBugs

<http://www.australianscience.com.au/research/google/34339.pdf> -exp using findBugs Ref Later

<http://www.cs.umd.edu/~jfoster/papers/issre04.pdf> -comp of static ana tools Ref Later

FindBugs documentation is a bit behind from their lastest implementation.

FindBugs uses a syntatic bug pattern detection to detect bugs[http://www.cs.umd.edu/~jfoster/papers/issre04.pdf]. FindBugs syntatically matches source code to known suspicious programming pratice [http://www.cs.umd.edu/~jfoster/papers/issre04.pdf]. This will try see where certain parts of code are used and check if it is applied in the correct manor. Findbugs also uses dataflow analysis to check for bugs [http://www.cs.umd.edu/~jfoster/papers/issre04.pdf].

FindBugs can make use of data flow anaysis and generate a control flow graph, this shows information such as Loops, If statements [ http://www.australianscience.com.au/research/google/34339.pdf]. It means that FindBugs can then see things such as Unitalized variables through an if statement which can be done via most IDE’s anyway or derferancing a null pointer. There may be a branch that does not give a variable a value, that will be picked up by findbugs as, correctness issue.

FindBugs very much does the same job as PMD, in their implementation they have a GUI just as PMD and have a rank on errors, Ranging from scariest to concern to alert the programmer how severe that bug could be. When google used FindBugs they found that 75% of issues were classified as “must fix, should fix or I will fix”[ http://findbugs.sourceforge.net/]. This shows that the bugs picked up by FindBugs are actual bugs that can break code.

It reports bugs with possible severity and a possible explanation as to why the bug is happening, this is all expressed via the GUI as with PMD.

Features of FindBugs are mainly to do with Control Flow, Type Checking, Null Checking,Dead Code and performance checking, derferancing a null Pointers and can find Infinate Loops in code.

One report shows that FindBugs cannot detect null pointer dereferences that occure only if a particular path through the program is executed, as the anaysis does not know if that path is feasible [ http://www.australianscience.com.au/research/google/34339.pdf].

# 2.2 Usage in the Industry

Languages like spec# and Spark are mainly used to develop high integrity systems, this find of development needs to be secure and free of errors. Although not all errors can be eradicated using languages that offer static analysis, it can go a long way to solve problems that may have led to problems.

These languages evaluate the program in such a way that a programmer could not without serious thought and a lot of testing.

Some examples of Systems that use these types of languages.

# 2.2.1 SHOLIS

The SHOLIS system was used to improve safety information on take-off and landing [http://www.sigada.org/conf/sigada2000/private/SIGAda2000-CDROM/SIGAda2000-Proceedings/Chapman-Presentation.pdf].

The system defines safe operating envelopes for prevailing wind conditions and sea states [http://ai2-s2-pdfs.s3.amazonaws.com/6cce/d507a9f1b42c96f842924c94a60911dc28c4.pdf]. The SHOLIS system made a lot of use of sparks tools, which was being used to meet the requirements of the UK Defence standard [http://ai2-s2-pdfs.s3.amazonaws.com/6cce/d507a9f1b42c96f842924c94a60911dc28c4.pdf]. This shows the effectiveness of the tools. Being able to prove that a program complies fully against the specification means a more reliable program, but only if the specification is correct in the first place.

Ada’s specification type of development helps aid this process, by defining behaviour in the spec it can be translated into source code. Spark would alarm the user if the source code does not comply with the specification. Using formal design The SHOLIS project could prove code directly against the specification [http://www.sigada.org/conf/sigada2000/private/SIGAda2000-CDROM/SIGAda2000-Proceedings/Chapman-Presentation.pdf]

The Document explaining the SHOLIS system says that it was mixing different critically levels of software in a single memory space along with running a single processor, showing how efficient Spark is [http://ai2-s2-pdfs.s3.amazonaws.com/6cce/d507a9f1b42c96f842924c94a60911dc28c4.pdf]. During testing, Spark tools can be used to statically verify code. If then code passes the checks set by spark, then the code for pre’s and posts can be removed to save performance during runtime.

In this study, they proved, freedom of runtime exceptions, partial correctness and safety properties. They found that this form of testing was more cost effective and concludes that it as more cost-effective than unit testing [http://www.sigada.org/conf/sigada2000/private/SIGAda2000-CDROM/SIGAda2000-Proceedings/Chapman-Presentation.pdf], although unit testing or some other form of testing should be used on parts that are identified as non-spark code.

The proofs used discharged 9000 verification conditions, which is components of Sparks verifier which uses proofs to discharge verifications using formal verification methods eg Pragmas in spark such as assert can help tools prove a condition.

This document shows the Spark and Ada combined collectively make up checks, due to the spec nature of Ada it is easier than most languages to prove the specification, it also shows Ada’s and sparks performance, being able to run an intensive program with one core, with the switching of software such as logging.

# 2.2.2 Lockhead

Lockhead is aircraft mission computer, it generated 130,000 lines of safety related code [http://www.sigada.org/conf/sigada2000/private/SIGAda2000-CDROM/SIGAda2000-Proceedings/Chapman-Presentation.pdf]. The mission computer had to overcome some obstacles to be a success.

The Lockhead mission computer had to meet DO-178D which is a software certification for airborne systems [http://www.adacore.com/gnatpro-safety-critical/avionics/do178b/].

The document contains a few guidelines to meet the certification including verification of airborne software. The certification implies that if met then there is a certain degree of confidence that the software will work safely, and if not the side effects that an error will have on the flight.