International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 5, May 2015, PP 1-12
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Evaluating Dependability and Performance of Programing Languages for Critical Systems

Mughele Ese Sophia

Department of Computer Science.Delta State School Marine Technology, Burutu, Nigeria,
prettysophy77 @yahoo.com, prettysophy99@gmail .com

Longe Olumide Babatope (PhD)

Department of Computer Science, University of Ibadan, Ibadan, Nigeria. *longeolumide@yahoo .com*

Abstract: *Ability for critical systems to perform optimally is a major concern for both the customer demanding for such system and the software engineer. This paper takes a critical look at the term "critical system", the types, and its sensitivity. Hence, if a critical system does not perform as expected it will impact seriously on life, environment and valuable assets. To ensuring an optimal performance of a critical system, there is underlying software that enhances its performance. The most emergent property of a critical system is dependability which is a function of the software used in designing the system. Dependability in a system implies that the system consist of the following functions; availability, reliability, safety and security. This paper focuses on the programming language support for dependability and performance for critical system. To what level does each of the language discussed in this paper support dependability of critical system and how possible is it to achieve dependability as well as performance. Finally, this paper also proposes a language rating for dependability of critical system. This will enable software engineers to evaluate this rating to make a proper choice when developing software for critical systems.*

Keywords: *Software development, Programming language, Critical systems.*

1. INTRODUCTION

Critical Systems are systems whose breakdown or decline in performance can cause threat to human life, substantial economic losses or physical damage. Critical systems are socio-technical or technical systems that businesses or people require. If these systems fail to deliver their services as expected then serious problems and significant losses may result (Somerville, 2004). According to Critical system Lab, (2009), they opined that Critical systems are systems that could cause a drastic effect on both human life and property when a defect occurs. It is expected that critical system satisfy a variety of security, safety, availability and reliability. Three main categories of critical systems exist;Mission-critical system is one whose breakdown may cause the breakdown of some goal directed activity.Examples of MCS includespacecraft navigational system for, rocket; Business-critical systems which are systems whose breakdown may causevery high cost for the business using that system, an example of BCS is software accounting system in Banks; andSafety- critical system is one whose breakdown may causephysical damage,severe environmental damage and loss of life. Examples of SCS include control system for chemical manufacturing plant, Nuclear plant (Somerville, 2004).

A life-critical system or safety-critical system is one whose breakdown or malfunction may lead toloss or severe damage to equipment or environmental harm, death or serious injury topeople. Some common domains where critical systems are applicable are examples of a software-based system for controlling space ship, and software used in processing medical images for treatment planning and ailment diagnoses. These critical systems functions with the underlying activities and operation of embedded software, called safety critical software, the software has various functions, and these functions enable the safe operation of the critical systems. The function of safety critical software includes the following: Safety critical software implements a crucial decision- making process, They

are software that regulate and monitor functions that are safety critical, Safety critical software acts as protective agent when risk occur in the system. They are software that impacts systems that run safety critical software, software that runs on the same target system as safety critical software, they validate and authenticate safety critical software.

Critical software may or may not be embedded. Since critical systems are systems that can result in loss of life, health and even cause hazard to the environment when errors occur hence, there is need to put into consideration factors that will enhance the implementation of programming techniques that will be used for building dependable software for critical systems. Dependability and performance becomes a key issue in the design of a critical system. Dependability of a critical system can only be achieved if the following components are present in the design of the software; this is known as dependable software development for critical systems.

2. RELATED LITERATURE

2.1 Programming Techniques for Developing Dependable Software Systems

The techniques deployed for software development must guarantee Software dependability. Precisely, software end user expect software product to be dependable no matter the cost. Nevertheless, for noncritical systems, end usermay be ready to allow some level of system malfunction. Some software, have very high dependability requirements and better programming procedures will ensure these requirements are met. To achieve dependability the software system must consist of the following elements.

Error Detection: This is a method whereby thesoftware is developed to prevent human error and to minimize system defect, the system is coordinated so that defects in the software are perceived and corrected beforebeen delivered to the user.

Failure Endurance: The system is created in such a way that defects in the software does not cause system failure. Incritical situations, software systems need to be tolerate failure. Failure Endurance is needed where system failure costs are very high orthere are high availability requirements.The system must also be fault tolerant even if the system is fault-free (Somerville, 2000). Fault tolerance provides the system with the following action components.

Methods to failure accommodation

Defensive programming; There is always assumption by programmers that defects exist in the program and add backup codes to confirm the condition after codifications to make sure that it is coherent. Failure- tolerance architectures; software and hardware system architectures that support software and hardware duplication and a failure accommodation controller that perceive difficult and permits failure recovery. Steps in Fault recovery Forward recovery is a method appliesrestoration to an erroneous system condition.

Backward recovery; it restoresthe system state to a known safe state.Backward error recovery is easy and simpler. Every facts of a safe state are maintained and this replaces the Corrupted system state of Forward recovery. In backward recovery, transactions are a frequently used method of backward recovery. Modifications are not applied until computation is complete. If an error occurs, the system is left in the state before the transaction.

2.2 Evaluation Criteria that Makes a good Language

1. Readability - The code is easy to read and understand (What makes this next)

1. Language is appropriate to problem (if not the solution may be unnatural and odd, for instance a language that does not have addition property perform addition function)

2. No Tricks or Puzzles (doing things more than one way for a situation)

3. Structured control, There is no go to's , the form of the code should be close to 1 for execution

4. Syntactic Consistency - if it looks different, then it should execute differently in a consistent manner

2. Simplicity - easy to use and learn

1. A small definition size for language constructs so all programmers know the same material

2. NO feature multiplicity that is you can only do something one way; EX: not like c++ incrementing (i++, i=i+1, i+=1)

3. Operator overloading is not allowed because the definition of the overload is dependent on the programmers meaning

4. Natural notation - EX: reads left to right, addition is in equation form not something looking clumsy.

5. Regularity of Notation rules for something will always be the same, never be the same, or are the same one time (people have a hard time remembering anything beyond that / special rules)

6. Extreme simplicity can be a hindrance as can be the case with assembly language

3. **Orthogonality**, without orthogonality the language will have rule exceptions while, high orthogonality however will lead to general rules which can be used in different instances. EX: a pointer should be able to point to all data types, not just integers;

1. Too much of Orthogonality is also bad as because there must be a basic construct to match the larger number of definitions, therefore the basic constructs are more numerous leading to higher numbers of definitions

4. **Syntax Design**

1. All identifiers should be able to have long names that are identifiable to function they perform. 2. Special Words should be defined clearly, if the special words are too general then it is hard to tell the role they are playing; EX: } then end of a block in C++ could be for an if or a loop. a. ALSO: Special words should be restrictive and not used for variables

3. Form and Meaning should be uniform not dependent on case usage

5. **Writeable**, this is the ease of creating a program for a specific problem(domain of the language)

1. Adequate built in types and operators

2. Support for abstraction by extending the definitions and operators

3. Ignore Implementation because the code will work the same on two different machines

4. Support for parallel programming and other paradigms; EX - java can only do object oriented writing which may not be the most simplistic way of doing something

6. **Reliability**, if the program performs as expected under all conditions it is reliable (it returns an information if it did not perform as expected)

1. Type checking program for type errors (an integer cannot suddenly become an array)

2. Private Boundaries are enforced

3. Exception handling it helps to stop run time errors

4. There is no aliasing because it relies on the programmer ability to recall when to change values

7. **Portable**, it has been standardized preferably by a standardizing organization like ANSI or ISO

1. Has lots of compilers for different hardware but will run the same

8. **Re-use of Code**

1. Libraries of functions

2. libraries of data structures

1. That are general as is the case of template in C++

2. Inheritance - using what you want from a library and adding something specific

9. **Cost** - the total cost must be reasonable for its functional use

1. Cost of training programmer

2. Cost of writing the program

3. Cost of compiling the program (some compilers are very expensive)

4. Cost of executing program - if the run time to start the program is too long users will not buy it

5. Cost of implementation - does the program only run on expensive hardware

6. Cost of poor Reliability - a failure in a critical system can cause lawsuits etc. EX: a missile is launched and hits a home because its guidance program failed

7. Cost of Maintenance - if the program must be watched daily it will require personnel (Programming Language Concepts, 2010).

2.3 Influences on Language Design Beyond Evaluation Criteria

1. **Computer Architecture** - Most of the computers are based on the Von Neumann Architecture and as a result the languages are imperative; Thus recursion is not as efficient and not used as often.

2. **Programming Design Methodologies** - As programmers refine their needs to create programs, languages have changed. EX: top- down design and data-oriented program design (data abstraction)

3. **Language Categories** - imperative, functional, logic, and object oriented (Some people consider this a growth from imperative languages and therefore does not have its own category)

4**. Language Design Trade-Offs** - what is more important in the program between the criteria for a good language; EX: Write-ability may fall as the Reliability increases

5. **Implementation Methods** - If one processor supports a command and another does not then code using that command will not run on both machines

6. **Compilation** - creates object code that is relevant to the machine 1. Process as follows |

Source Code > Lexical Analyzer > Syntax Analyzer > Semantic Analyzer >CodeGenerator > Object File (Program Executable)

1. **Source Code** - Program in High Level Language

2. **Lexical Analyzer** - Checks symbols being used in the source for correctness; EX: legal names, order not specific

3. **Syntax Analyzer** - AKA parser | Checks the grammar rules of the language; EX: for loop setup correctly

4. **Semantic Analyzer** - Checks the meaning of constructs; EX: Type mismatch error

5. **Code Generator** - Translates code to related assembly / machine code

6. **Object File** - File that can be run on machine 2. Advantage is speed during runtime

7. **Pure Interpretation** - checks each line of code and runs it one simulation machine (Virtual Machine) (Programming Language Concepts, 2010).

1. Advantage is debugging during runtime

2. Much slower than compiled code.

With the above listed components of dependable software development for critical systems, there is therefore, need to determine the programming language that will support the dependability and performance of software development for critical systems. The most emergent property of a critical system is dependability. This term was proposed by Lupric (Lupric, 1995) to cover the four major system attributes that are related. These are reliability, availability, safety and security. To achieve dependability, this is the most important emergent property of critical systems. It is also important to note that the language support that will enhance system dependability cannot project high performance because in critical system, dependability and performance has inverse relationship. The

higher the dependability level, the lower the performance rate of the system. Some programming languages can support dependability of critical system but the level of support varies from one programming language to another.

2.4 Language Support for C++ for Dependability of Critical System

C++, originally named C with Classes,was developed by Bjarne Stroustrup in 1979 at Bell Labs as an improvement to the C language. It was renamed C++in 1983 (Stroustrup, 2010).C++ is a statically and also a free-form, compiled, multi-paradigm, typed, general-purpose programming language. It is considered as an intermediate-level language, as it consist of a combination of both low- level language and high -level language properties(Schildt, 1998).

Of all programming languages ever developed, C++ is one of the well-known and the domains it is being applied are but not limited to the following; embedded software, device drivers, apps, systems software,client applications, and high-performance server.It is also applicable with entertainment software such as video games (Programming Language Popularity, 2009) and (TIOBE Programming Community Index, 2009).Both open and privately owned compiler software for C++ are supplied by organisations, such as the Microsoft, Intel, Embarcadero Technologies and GNU Project. C++ has immensely affected positively the development of several programming languages of this day, which are Java and C-Sharp (C#).

C++ programminglanguage started as improvements to C, first including classes, then virtual functions, multiple inheritance, templates, overloading of operators, as well as handling of exceptions, etc. The standards for C++ programming language, was formally approved in 1998 as (ISO/IEC 1998). The 2003 technical corrigendum amended the C++ standard, known as ISO/IEC 14882:2003. C++0x (still informal), the next standard version is still under development.C++ isnow greatly adopted in the creation of software for embedded systems, hard-real- time systemsand safety-criticalsystems. Other languages may be better used for safety critical systems design, but other significant factors favour C++ such as tool support availability and developers that are highly skilled.

Therehas definitely been larger attention towards the use of C++ in the design of safety-critical systems since its use in the creation of air vehicle software for the Joint Strike Fighter (JSF), and the well-known release of the standard for coding C++ used in the SF C++ project.

Recently,(i.e. June 2008) the standard for MISRA C++ "Guidelines for the use of the C++ language in critical systems" was released by the Motor Industry Software Reliability Association (MISRA) which is identical to the JSF C++ standard.The standard also stipulates where both differ. For instance, JSF C++ prohibit using C++ exceptions, while MISRA C++ stipulates emphatic rules usage. Although, C++ (as well as C), by its development, not genuinely a right language to write high integrity or safety- critical software, C++ grants a programmer much freedom(ISO/IEC, 2003). With the freedom comesa lot of responsibilities, nonetheless,sufficient justification for its use in safety-critical system:Such reasons and functions of C++ makes the support of dependability for critical systems, though there is a level to which C++ can support critical systems. The following reasons make C++ malleable and dependable for critical systems:

1. **Better level of abstraction**: C++ provides a better level of abstraction than C. This makes the language more fitting for larger complex applications. Moreover, low-level programming necessary for high-performance code or hardware access is allowed.
2. **Tool support**: Many tools that permit model- driven development can automatically create C++ code automatically.
3. **Portability**:The extent to which C++ compilers are widely available for nearly all hardware platforms provides easier porting of applications to new or lower-cost processors at any period in a project.
4. **Object-oriented programming:** C++ supports object-oriented programming paradigm, which is efficient in modern software development projects**.**
5. **Efficiency**: C++ compilers creates code that performs efficiently, or even more than C code.
6. **Maturity:** C++ is fully-developed, well-analyzed and proven to be good in practice.

7. **Skilled developers:** Since C++ is one of the most greatly used programming languages, there is availability of a lot of developers C++ that are highly skilled.

Nonetheless, there are many concerns that restrict the suitability of C++ for safety-critical systems, these include:

1. C++ is a highly complex language. C++ needs an enormous effortto learn and to fully understand.This also make the likelihood of compiler errors to increase.C++ code doesn't at all-time behave as an individual would 'instinctively' confidently believe from looking at the source code.
2. Programmers make mistakes that cannot be identified by a compiler (for instance, using assignment operator '=' in place of the comparison operator'==').
3. Issues of portability arisebecause the semantics of some C++ constructs are not completely stipulated. Hence, the behaviour of a program may differ(unspecified,undefined and implementation- defined behaviour).
4. C++do not ensure built-in run-time checks (for instance, array bound errors orarithmetic overflows.
5. C++ permits too many flaw to get around the typed system unintentionally or deliberately, since it is a strongly-typed language. Programmers must avoid any language constructs and code that can possiblycausestrange program behaviour. C possess these concernstoo but this has not stopped its adoption safety-critical systems design. This is due to coding standards adoption which restrictfeatures of language to a safe subset.

2.5 Standards for Coding C++

MISRA-C "Guidelines for the use of the C language in critical systems" isthe most appealing standard coding C in safety-critical systems (MISRA-C,2004). It wasfirst published in 1998 and then revised in 2004, it stipulates a "safe" subset of the C language in the form of 121 required and 20 advisory rules. MISRA-C greatly influenced another standard for coding with C++ in safety- critical systems.It benefitted from great acceptance among safety-critical applications developers, which is beyond the automotive industry which was the initial target.The "Joint Strike Fighter Air Vehicle C++ (JSF C++) Coding Standards for the System Development and Demonstration Program" (Joint Strike Fighter, 2005), was new and different, this signaled diversion from Ada as the authoritative programming language to develop avionics software by US Department of Defence. There is a bit difference in principle of JSF C++ and MISRA-C (and the new MISRA-C++) because coding style and metric guidelines are defined, which is absent in MISRA standards. For instance, AV Rule 1 needs that "any one function (or method) will contain not more than 200 logical source lines of code", and AV Rule 50 needs that the first word of the name of a class,enumeration, namespace, structure, or type created with typeset will start with block letter. After which all others will be lowercase. JSF C++ states 221 rules in total. Thepublication of MISRA-C++ was due tothe tendency to develop critical systems with shift C to C++ (David, 2008),in 2008. MISRA-C++ adds many more C++ specific rulesand takes many rules from MISRA-C: 2004,making it 228 rules in total.

2.6 Factors That Make C++ Support Dependability of Critical Systems

There are many facilities provided by C++ that make it easy code correctly and robustly. Though, standards for coding do not definitely recommend or enforce using these facilities, they can recommended generally. For instance, the Resource Acquisition Is Initialization (RAII) (Lockheed, 2005) idiom aid to implement and manage resources correctly. The requirement of smart pointers are provided, which makes errors related to pointeravoidable (MISRA-C: 2008).

RAII also implements scoped locks which makes implementing critical sections correctly easier. Templates also make writing robust C++ code easier,which enable the implementation of bounds-checked arrays. The implementation of fixed- point arithmetic isan interesting use for templates in safety- critical Systems.A C++ developer should handle exceptions well (David, 2008). Error handlingis greatly simplified with exceptions. RAII is highly efficient and effective tool to write exception safe code providedthat the right standard for coding such as MISRA-C++ or JSF C++ is met.JSF C++ and MISRA -C++ standards are compared, thus, similarities, conflicts as well

as differences are shown.Programmers should familiarize themselves with the details of the standards in safety-critical design.

2.7 Level of Support of Java Language for Critical System

Highly efficient predictable code execution is required Real-Time/Safety-Critical systems design. Techniques to achieve this include the use of RTSJ-compliant Virtual Machine, Ahead-Of-Time compilation and incremental Garbage Collection. Moreover, time-predictability can be destroyed without difficulty with standard library usage (array resizing, lazy initialization, etc.). Achieving true time predictability, a time deterministic libraryis required (Marlborough, 2007). Java language is very sophisticated for the enhancement of dependability for critical systems. Improved reliability, productivity and scalability. This makes development of mission-critical applications switch from Ada/C/C++ to Java (COTS Journal, 2006). "Real-Time" means "capable of simulating a process at a rate that matched that of the real process itself" and "reaction to an event within a strict deadline".

Recently, developers are using java to develop critical systems such systems such as, NASDAQ stock exchange for instance use real- time Java for prototypingsystem (At JavaOne, 2007).DDG-1000 also developed bythe militaryis also "powered" by real-time Java software. Though, the strength of the real- time "chain" is as strong as its weakest link. Still, there is one major issue which, is the standard library. The language allows for real- time garbage collection, ahead-of-time compilation, a highly time deterministic operating system the issues. Java language has some level limitations this may affect its dependability for critical systems because the standard Java library isn't time-predictable and a real- time Java environment would contain time-deterministic library like the multi-platform Javolution solutionin order to enhance dependable support for critical systems.

Apparently, it could argued that Java's strength lies in its all-encompassing standard library which are "components for graphical user interfaces, database access, networking, XML processing, logging and many other areas". The problem is thatsafety critical systems are not part of the intentions of these components. Till date,Sun Java license stipulates definitely that Java shouldn't be used for the operation of "Nuclear facility(Sun Microsystems License Agreement).In lots of instances, Javolution classes are faster than their standard library counterparts, which proves that real-fast and real-time can both be achieved. Good-performance, simplicity, and bounded-response-time clarifyJavolution project success when combined and the reason its current adoption by developers and reputable organizations (Sun, Excelsior, Raytheon, IBM, Lockheed Martin, Blockbuster, Thales, BEA, etc),(Marlborough and Dautelle, 2007).

2.8 Language Support of Ada for Dependability of Critical Systems

From 1977 to 1983, Ada was innovatively developed by a groupguided by Jean Ichbiahof CII Honeywell Bull in formal agreement with United States Department of Defence (DoD).Ada language is an object-oriented, commanding, structured and statically typed high- level language, gotten from Pascal and other languages. The language have a robust standard language support for definite concurrency, synchronous message passing, offering tasks, non-determinism and locked objects.Due it's strong language construct it superseded the several programming languages then used by the DoD. In mission-critical-applications, "such as avionics software", it is a robustly typed and validation is done for compilers for reliability. Joint ISO/ANSI standard defines Ada 2005 which is a current version (ISO, 1995), together with major Amendment ISO/IEC 8652:1995/Amd 1:2007(ISO/IEC 8652, 2007).

The target of Ada innovatively was real-time and embedded systems. The Ada 95 revision, was developed by S. Tucker Taft of Intermetrics in 1992 to1995, it was used for improved support for systems, financial, numerical, and "object-oriented programming (OOP)". Some distinguished characteristics include "modularity mechanisms, strong typing, parallel processing (tasks, synchronous Message passing, protected objects and nondeterministic select statements), run-time checking, genericsand exception handling".its syntax is simple, readable and consistent. Choices of ways to perform basic operations was minimized, and English keywords (e.g. "or else") was preferred to symbols (e.g. "||"). Basic mathematical symbols (such as.: "+", "-", "*" and "/") are used for operations while other symbols are avoided**.**

Ada is developed for of very big software systems. Detection of problems early during the design phase is possible before implementing due to the fact that there is separatecompilation of Ada package specifications ("the package interface") without consistency checks. It supports compile-time checks which make detection of bugs avoidable which would not be detected until run-time in some other languages. For instance, the syntax needs definitely named closing of blocks for errors prevention which can be caused by mismatched end tokens. Ada's concurrency makes it possible to diagnose deadlocks. Compilers also check for the following; "visibility of packages, misspelled identifiers, redundant declarations, etc." and indicate warnings and suggest how the errors can be fixed.Facilities that help program verificationcan also be included. Due to these,Ada is greatly used in "critical systems", where any irregularity might cause very severe result, (e.g., injury, accidental death or severe financial loss). Systems that use Ada railways, military, avionics, space technologyand banking (Tucker and Olsen, 1999) and (Feldman and Michael).

Ada's dynamic memory management is type-safe and high-level. Ada doesn't have general (and unclear) "pointers"; nor does it absolutely state any pointer type. Thus, all allocation and de-allocation of dynamic memory take place via definitely indicated access types. Each access type has a connected storage pool which deal with the low-level parts of memory management; the default storage pool or define new ones can either be used by the programmer.Ada permit a restricted form of region-based storage management;because a storage pool destruction also affects all the objects in the pool,using storage pools creatively can make for a restricted form of automatic garbage collection.

The Code for complex systems is usually maintained for years, by programmers other than the first developer. Design principles are relevant to most software projects, and software development phases, but in application to complex, safety critical projects, advantages in maintainability, reliability and correctness have priority over costs in the initial development.The Ada language definition "the Ada Reference Manual or ARM, or sometimes the Language Reference Manual or LRM" is free contentunlike most ISO standards. So,it is a general reference for both programmers implementing Ada compilersand Ada programmers.Therefore, it is important to note the Ada was originally developed to support critical system.

Consequently, Ada's is now used beyond military applications, such as commercial projects where a software bug can have severe result, e.g. air traffic control and aviation, commercial rockets (e.g. Ariane 5 and 4), satellites and other space systems, banking, and railway transport (Feldman, Michael). For instance, the fly-by-wire system software in the Boeing 777 was Ada-written(Norris and Wagner, 2001).The Canadian Automated Air Traffic System was written in 1 million lines of Ada (SLOC count) (Microsoft, 2011).The UK's next-generation Interim Future Area Control Tools Support (iFACTS) air traffic control system is designed and implemented using SPARK.AdaCore, (2007), noted that the FrenchTVM in-cab signalling system on the TGV high speed rail system, and the metro suburban trains in New York City, London, Paris and Hong Kong are implemented using Ada(Feldman, Michael) and (AdaCore,).

(Pyle, 1991) and (Cullyeret. al., 1991) opined that no language alone can ensure safety-critical systems construction i.e. additional formal methods are needed. Nonetheless, Ada's support for system reliabilitymakes it comes closer than other existing languages by an order of magnitude (Pyle, 1991). Ada 95 has a Safety and Security Annex. Rating: 6 in tab. 1. The under listed points judges Ada 95 with respect to the language criteria presented for a good Languagewhich makes it dependable for critical systems

- Independent definition of any particular operating system or hardware
- Standardized definition and standard compliant compiler implementations
- Software engineering technology support, poor practices prohibition, and maintenance activities support.
- Effective and efficient support of application domain(s) of interest.
- Level of system reliability and safety support.
- The current technology state should commensurate with implementations of compiler.

2.9 Language Support of Visual Basic (VB) for Critical Systems

Visual Basic is a very easy language for new developer and it is a very basic programming language. The language can be learnt easily compared to other languages. It consists of all most all the basic text

and graphical related features. It enables access to all popular database systems. Using VB program, the components can be arranged on a form by mentioning the attributes of each component. More also, multiple components can be arranged and also be made as a single executable file for software. The basic drawback with VB is that, some aesthetic features cannot be incorporated with VB this setback makes the language unsuitable for the dependability of critical systems.

2.10 Language Support of Visual Basic .NET for Critical Systems

VB.NET is quit advanced compared to Visual Basic. This programming language has integration usually with Microsoft .NET framework. More aesthetic features can be incorporated for the software developed using Visual Basic .NET compared to simple Visual Basic. The coding complexity here is high compared to Visual Basic. Nevertheless, performance of a software developed using VB .NET is better compared to VB. Though V.B. NET is more improved than V.B yet it is not a very competent language for critical systems.

2.11 Language Support of C Sharp (C#) for Critical Systems

C Sharp is an Object Oriented Programming Language like VB .NET. It is easier to handle data type support and memory allocation effectively compared to Visual Basic and Visual Basic .NET. The language is more appropriate for large size embedded system applications and complex software. Though the aesthetic features that can be achieve with C# is varied with Visual Basic .NET, comparatively it is difficult to program with C#. Hence it is specifically not recommended for medium size programs. It can be concluded that for medium size software applications, Visual Basic .NET is the best language to develop software application provided aesthetic feature is considered as priority. Otherwise, one can program using Visual Basic. Therefore, V.B, V.B. NET, and C# were not originally designed to support critical systems.

3. RATINGS OF COMPUTER PROGRAMMING LANGUAGES

According, to (Lawlis, 1997), various known programming languages on a scale of zero to ten for each of the fourteen criteria.Scripting languages such as Perl and Python, were not rated but it does rate assembly language just for reference purpose. The results appear in the Table below. The fourteen divisions are briefly explained below the Table, but they are listed here for reference purpose: A. Clarity of source code B. Complexity management C. Concurrency support D. Distributed system supportE. Maintainability F. Mixed language support G. Object-orientedH. Portability I. Real-time support J. Reliability K. Reusability L. Safety M. Standardization N. Support for modern support methods Table1 below shows Lawlis ratings for programing language support for critical systems. The rating evaluation is based on the above fourteen characteristics, and the table below shows that Ada has the highest rating for dependability support for critical system and Assembly language has the list support. This evaluation is based on Lawlis ratings of Computer Programming Language support for critical systems (Lawlis 1997). Tab1.Shows ratings of Computer Programming Language support for critical systems.

Tab.1. *Extracted from Lawlis 1997*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Ada 95	**9**	**9**	**8**	**5**	**9**	**8**	**10**	**8**	**7**	**9**	**8**	**6**	**10**	**9**
Java	**8**	**7**	**7**	**9**	**5**	**5**	**10**	**9**	**0**	**8**	**8**	**4**	**8**	**9**
C++	**6**	**6**	**0**	**0**	**7**	**7**	**10**	**7**	**7**	**5**	**8**	**3**	**5**	**7**
C	**5**	**5**	**0**	**0**	**2**	**5**	**0**	**5**	**7**	**1**	**3**	**0**	**5**	**1**
Fortran	**5**	**4**	**0**	**0**	**2**	**5**	**0**	**3**	**5**	**1**	**3**	**0**	**5**	**1**

Smalltalk	**9**	**6**	**2**	**0**	**7**	**3**	**10**	**3**	**0**	**3**	**8**	**0**	**3**	**7**
COBOL	**7**	**2**	**0**	**0**	**2**	**0**	**0**	**3**	**0**	**3**	**3**	**0**	**5**	**1**
Assembly	**1**	**2**	**2**	**0**	**0**	**0**	**0**	**1**	**5**	**0**	**1**	**0**	**0**	**0**

A. **Source code clarity** - degree to which basic language properties ensure source code that is readable and understandable and that clearly shows the underlying logical structure of the program.

B. **Complexity management (architecture support)** - the degree to which basic language properties enhance the management of system complexity, in terms of addressing issues of data, algorithm, interface, and architectural complexity.

C. **Concurrency support** - the degree to whichtheir intended functions in a satisfactory inherent language features support the manner throughout the expected lifetime.

D.**Distributed system support** - the degree to which basic language properties enhance the construction of code to be distributed across multiple platforms on a network.

E. **Maintainability** - the degree to which fundamental language properties enhance the construction of code that can be readily changed to satisfy new requirements or to correct deficiencies.

F. **Mixed language support** - the degree to which basic language properties ensure interfacing to other languages.

G**. Object-oriented programming support** - the degree to which basic language features enhance the construction of object-oriented codes.

H. **Portability** - the degree to which basic language features permit the transfer of a program from one hardware and/or software platform to another.

I. **Real-time support** - the degree to which basic language properties permit the construction of real-time systems.

J. **Reliability** – the degree to which basic language features permit the construction of code with several threads of control (also known as parallel processing).

K.**Reusability** – the degree to which basic language features permit the adaptation of code for use in other application.

L. **Safety** - the degree to which basic language properties permit the construction of safety- critical systems, yielding systems that are failure-tolerant, fail-safe, or robust in the face of systemic malfunction.

M. **Standardization** - the extent to which the language definition has been formally standardized (by identified bodies such as ANSI and ISO) and the degree to which it can be reasonably expected that this standard will be followed in a language translator.

N**. Support for modern engineering methods** - the degree to which basic language properties permit the interpretation of source code that strengthens suitable software engineering principles.

REFERENCES

[1]. AdaCore, (2007)."GNAT Pro Chosen for UK’s Next Generation ATC System". http://www.adacore.com/2007/06/19/adacore-gnat-pro-chosen-for-uk-next-generation/. Retrieved 09-03-2011.

[2]. AdaCore. (2008)"Look Who’s Using Ada" http://www.adacore.com/home/ada_answers/ lookwho/ . Retrieved 09- 03-2011.

[3]. At JavaOne, (2007), NASDAQ’s CIO took the stage to speak on their system, considered the fasted in processed per second.

[4]. COTS Journal, (2006): "Software Modernization Is Key to Controlling Costs, Complexity" July 2006

[5]. Cullyeret. W, Goodenough. S, Wichmann. B, (1991), "The choice of computer language for use in safety-critical systems," Software Engineering Journal, Vol. 6, No. 2, March 1991.

[6]. Critical system Lab, (2009) what are Critical Systemsen.wikipedia.org/wiki/Critical_system

[7]. David.A.(2008), Exception-Safety in Generic Components, http://www.boost.org/more/ generic_ exception_sa fety.html

[8]. Dautelle. J and Marlborough. R. (2007) Fully Time Deterministic Java™ Jean-Marie Dautelle1 MA, 01752. http://javolution.org/doc/AIAA-2007-6184.pdf

[9]. Feldman, and Michael. "Who's using Ada?".SIGAda Education Working Group. http://www.seas.gwu.edu/~mfeldman/ada- project-summary.html#Banking_and_Financial_Systems_. Retrieved 09-03-2011.

[10]. ISO/IEC 14882 (2003) International Standard: Programming Languages – C++, Second Edition, ISO/IEC, 2003

[11]. ISO/IEC 8652: (2007),1995/Amd 1:2007Retrieved 09-03-2011.

[12]. Javolution, (2007) (http://javolution.org) supports the J2ME, J2SE/J2EE platforms and GCJ (GNU Compiler)

[13]. Joint Strike Fighter (2005), Air Vehicle C++ Coding Standards for the System Development and Demonstration Program, Lockheed Martin Corporation,

[14]. Lockheed. M, (2005) Joint Strike Fighter Air Vehicle C++ Coding Standards for the System Development and Demonstration Program, Lockheed Martin Corporation, 2005

[15]. Lupric .J. (1995).Dependable computing concepts, limits, challenges.Proc., 25th IEEE Syposium on fault- tolerantcomputing.Paxadena, CA: IEEE, (Ch. 3)

[16]. MISRA-C: (2004), Guidelines for the use of the C language in critical systems, MIRA Limited, 2004.

[17]. MISRA-C: (2008), Guidelines for the use of the C++ language in critical systems, MIRA Limited, 2008.

[18]. Microsoft, (2011), "How Many Lines of Code in Windows XP?" Microsoft. January 11, 2011. http://en.wikipedia.org/wiki/Source_lines_of_code

[19]. Norris. G and Wagner. M. (2001).Boeing 777: The Technological Marvel. Minneapolis, Minnesota: Zenith Imprint. ISBN 0-7603-0890-X.

[20]. Programming Language Concepts, (2010) copyright www.notes4free.com Retrieved 16-01- 12

[21]. Programming Language Popularity (2009). http://www.langpop.com/.Retrieved-01-03- 2011.

[22]. Pyle .C. (1991), Developing Safety Systems, New York: Prentice-Hall, 1991.

[23]. Schildt.H. (1998-08-01).C++ The Complete Reference Third Edition.Osborne McGraw- Hill. ISBN 978- 0078824760

[24]. Sommerville, I, (2000), Dependable Software Development, www.slideshare.net/.../ dependable- software- deve... - United States 20 May 2007 – Dependable software development Programming techniques for building ... Software Engineering [6th Edition] Ian Sommerville ·

[25]. Stroustrup.B. (2010). "C++ Faq: When was C++ Invented".ATT.com.http://www2.research. att.com/~bs/bs_faq.html#invention. Retrieved 2010- 09-16

[26]. Sommerville, (2004), Critical Systems Development, Chapter 20 Part 7th Ed (www.cs.au.dk) Showing 21 - 2 of 23 Items http://www.slidefinder.net/s/sommerville/9828337/p2

[27]. Validating Java™, (2005), For Safety-Critical Applications - AIAA 2005-6812

[28]. Tucker.S and Olsen.F, (1999). "Ada helps churn out less-buggy code". Government Computer News. pp. 2–3. http://gcn.com/Articles/1999/06/30/Ada-helps- churn-out-lessbuggy- code.aspx. Retrieved 09- 03-2011.

[29]. TIOBE Programming Community Index (2009). http://www.tiobe.com/index.php/content/ paper info/tpci Retrieved 01-03-2011/index.html

[30]. Wei F. and Hauser. C, (2005), "A Real-Time Garbage Collection Framework for Embedded Systems". ACM SCOPE '05 2005''. portal.acm.org. http://portal.acm.org/ft_gateway.cfm?

id=1140392&type=pdf&coll=GUIDE&dl=GUIDE&CFID =151 51515&CFT OKEN = 6184618. Retrieved 11 – 03- 2011

[31]. What's CvSDL?" (2003). http://www.cvsdl.com/: cvsdl. http://www.cvsdl.com/. Retrieved 01-03- 2011. "CvSDL was introduced in 2003 as a C++ class framework with Verilog features that worked like a Verilog simulator 1 2 3 4 5 6 7 8 9 10 11 12 13