

Software Quality

Chris Greenough and David J Worth

*Software Engineering Group
Computational Science & Engineering Department
Rutherford Appleton Laboratory*

c.greenough@rl.ac.uk

What is software quality

- ❑ **Software quality is a very difficult term to define!**
- ❑ **The IEEE definition: Software quality is:**
 - ❑ The degree to which a system, components, or process meets specified requirements.
 - ❑ The degree to which a system, components, or process meets customer or use needs or expectations
- ❑ **The Pressman definitions:**
 - ❑ Conformance to explicitly stated functional and performance requirements, explicit documented development standards, and implicit characteristics that are expect of al professionally developed software.
- ❑ **These are very different: one focusing on the users the other focusing on the process.**
- ❑ **Both definition agree that quality comes through conformance to a requirement specification.**
- ❑ **Not particularly useful to CSED!**

Software quality continued.....

J Frankovisch (1997)

- ❑ **ANSI(American National Standard Institution) defines quality as the totality of features and characteristics of a product that bears on its ability to satisfy given needs.**
- ❑ **Then what is software quality? Suppose you receive a software product that is delivered on time, within budget, and which correctly and efficiently performs all its specified functions. Does it mean a high quality software? For several reasons, the answer may be "no". Here are some problems you may find:**
 - ❑ **The software may be hard to understand and difficult to modify. This leads to excessive costs in software maintenance. Software maintenance cost may be very expensive, for example, General Motors spends 75% of their software effort in software maintenance.**
 - ❑ **The software may be difficult to use, or easy to misuse.**
- ❑ **Software quality is defined as the degree to which software conforms to quality criteria.**

□ Quality criteria include but are not limited to:

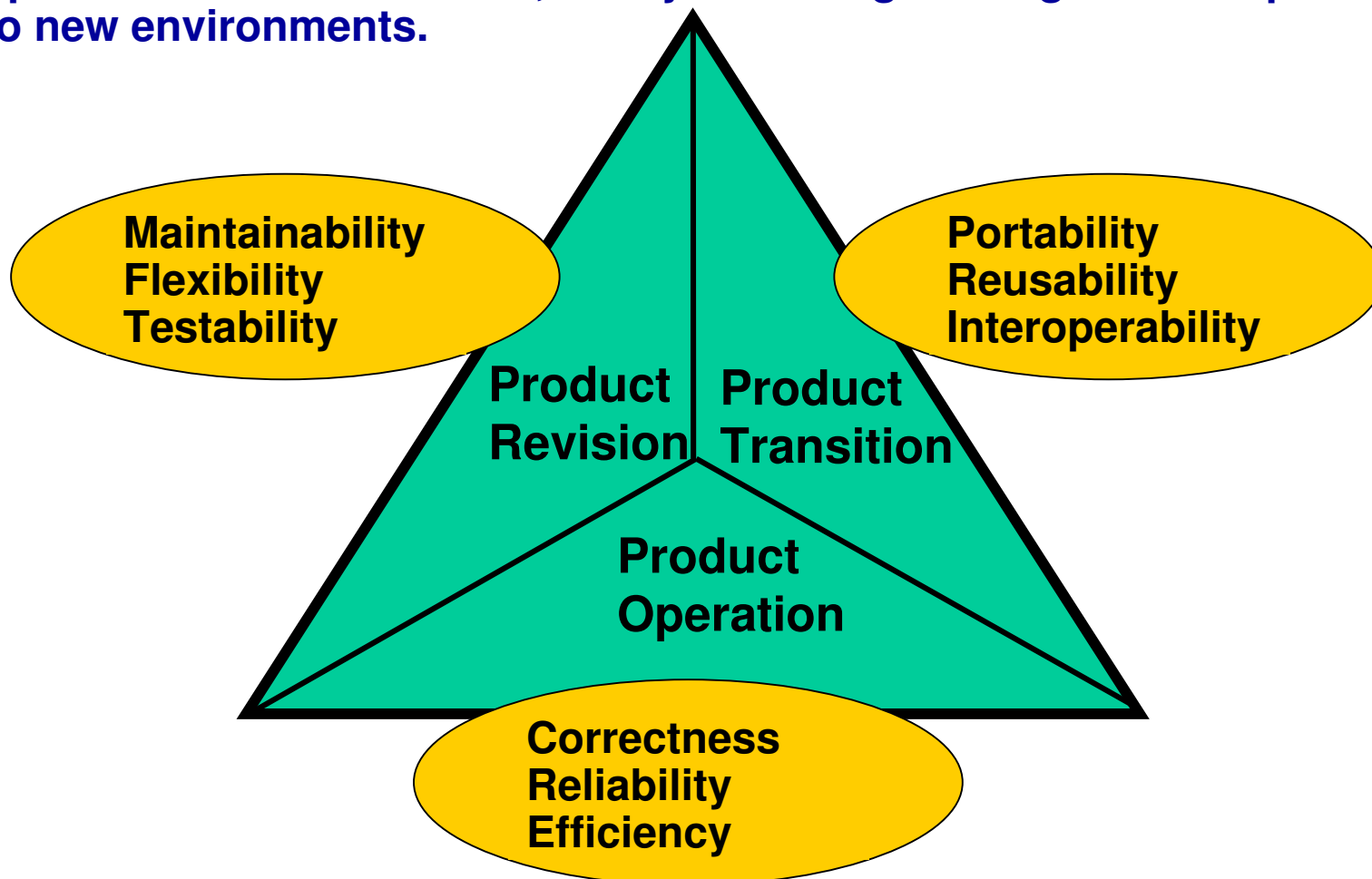
Economy	Correctness	Resilience	Integrity
Reliability	Usability	Documentation	Modifiability
Clarity	Understandability	Validity	Maintainability
Flexibility	Generality	Portability	Interoperability
Testability	Efficiency	Modularity	Resuability

□ Because there are so many criteria, we can not use them all for measuring software quality. But some of desired characteristics can be used in software product standards as, guiding actual development work to be done toward the right direction.

□ To be a high quality software, the product must be able to run correctly and consistently, have few defects (if there are), handle abnormal situation nicely, and need few installation effort. The defects should not affect the normal use of the software, will not do any destructive things to system, and rarely be evident to the users.

Measures of software quality

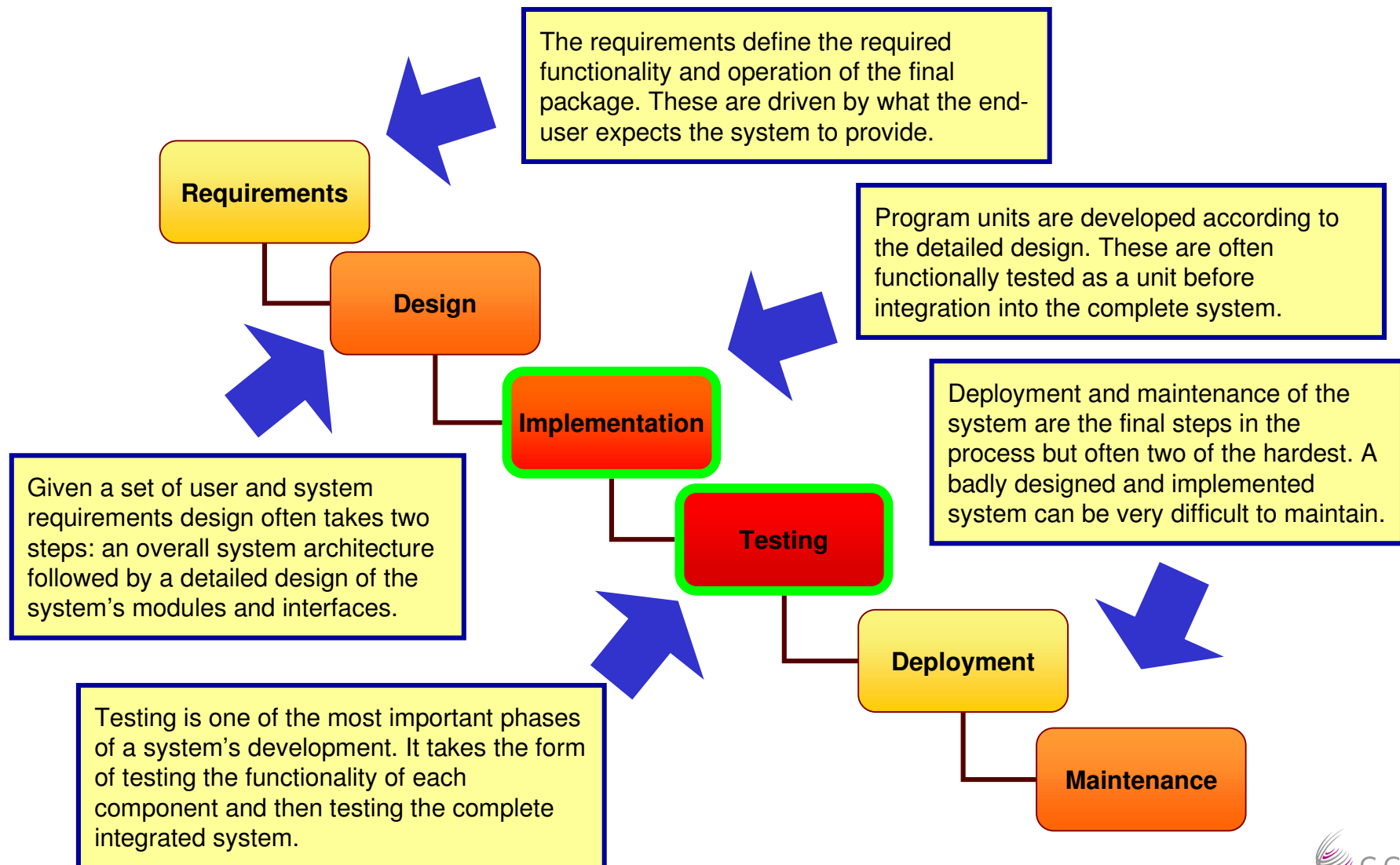
- **McCall's Quality Factors are based around three aspects of software: operational characteristics, ability to undergo change and adaptability to new environments.**



Basic characteristics of 'good' software

- High quality software must:
 - be reliable
 - be maintainable
 - be easy to operate
 - meet functional requirements of it domain

The Water Fall Software Life Cycle (one of many models)



FORCHECK for Software Quality and Standardisation

FORCHECK

Program Unit analysis

- syntax and type checking
- flags unused, unreferenced, undefined variables
- indicates truncation or loss of precision
- validates standards compliance

Global analysis

- verification of subprogram references
- verification of argument lists
- consistency of COMMON blocks
- verification of public module variables

Documentation

- cross-reference reports
- call tree

Running FORCHECK

- ❑ **Command line vs. interactive**
 - ❑ complete source vs. program unit

- ❑ ***forchk [global options] file [[local options] file...]***
 - ❑ can use wildcards - *.f
 - ❑ check conformance to standard
 - -f77, -f90, -f95
 - **compiler configuration file**
 - ❑ free format input
 - -ff
 - ❑ annotated source
 - -l *file*
 - ❑ report file
 - -rep *file*

Running FORCHECK cont.

❑ Environment variables

- ❑ FCKDIR install directory of FORCHECK
- ❑ FCKPWD password file
- ❑ FCKCNF compiler configuration file (default is Lahey f95 compiler)

❑ Potential Problem

- ❑ ordering of modules

❑ Solution

- ❑ create library
 - `forchk module_source_files -cre library_file`
- ❑ use library
 - `forchk [global options] file [[local options] file...] -lib library_file`

Outputs

❑ Errors

- ❑ (file: ../../model/singch.f, line: 292)
**[465 E] statement label expected

❑ Warnings

- ❑ *8
(file: ../../numerical/colnew.f, line: 466)
**[95 W] nonstandard Fortran syntax

❑ Information

- ❑ error1 .eq. resa
(file: ../../numerical/integ.f, line: 545)
**[340 I] equality or inequality comparison of floating point data
(comparing real data for (in)equality is potentially risky)

NagWare Tools for Software Quality and Standardisation

What are the NagWare Tools

- The NAGWare Fortran Tools provide users with the ability to analyse and transform Fortran 77 and Fortran 95 code.
- The NAGWare f95 Tools accept as input Fortran 77 and fixed or free format Fortran 95.
- Output from the transformational tools is always free format, so these tools are effectively fixed to free format translators.
- This set of tools provides analysis capabilities that include a call graph generator and transformational tools that include a configurable pretty printer, declaration standardiser and precision standardiser.
- These tools can be used in a range of ways:
 - Quality Assurance
 - Standardisation
 - Enforcing coding standards
 - Porting to new platforms
 - Converting from fixed format Fortran 77 to free format Fortran 95
 - Normal day-to-day development

f95 Tools Contents Summary

Analysers

- nag_coverage95 Source code coverage instrumenter
- nag_depend95 Dependency analyser
- nag_fcalls95 Call graph generator
- nag_modules95** **Module builder**
- nag_xref95 Cross referencer

Transformers

- nag_cbm95 Common block to module converter
- nag_chname95 Name changer
- nag_decs95 Declaration standardiser
- nag_mkintf95 Interface builder
- nag_polish95 Polisher
- nag_polopt95 Polish options editor
- nag_prest95 Precision standardiser
- nag_struct95 Restructurer
- nag_ustan95 Use statement analyser

f77 Tools Contents Summary

Analysers

- | | |
|--------------------------------------|----------------------|
| <input type="checkbox"/> nag_fcalls | Call graph generator |
| <input type="checkbox"/> nag_fxref | Cross referencer |
| <input type="checkbox"/> nag_libdoc | Interface lister |
| <input type="checkbox"/> nag_metrics | Complexity metrics |
| <input type="checkbox"/> nag_pfort | Portability verifier |

Transformers

- | | |
|--------------------------------------|----------------------------|
| <input type="checkbox"/> nag_apr | Precision transformer |
| <input type="checkbox"/> nag_chname | Name changer |
| <input type="checkbox"/> nag_decs | Declaration standardiser |
| <input type="checkbox"/> nag_lvi | Local variable initialiser |
| <input type="checkbox"/> nag_polish | Polisher |
| <input type="checkbox"/> nag_plopt | Polish options editor |
| <input type="checkbox"/> nag_profile | Profiler |
| <input type="checkbox"/> nag_struct | Restructurer |

NagWare nag_modules95

Synopsis

`nag_modules95 [options] infile`

- `nag_modules95` contains the full NagWare f95 Parser – the compiler frontend.
- If a filename with no suffix is provided the tool will look for a file with the suffix `.f95`, and if that does not exist, for a file with the suffix `.f90`.
- If a file has the extension `.f90` or `.f95`, it will be assumed to contain free format Fortran code. If a file has the extension `.f`, `.for`, `.f77`, or `.ftn`, the file is assumed to contain fixed format Fortran code. The options `-fixed` and `-free` may be used to override this behaviour.
- Modules (and include files) are expected to exist in the current working directory or in a directory named by the `-I` option.
- Details found in *man* pages:
`man nag_modules95`

nag_modules95 options

- free**
 - Forces the tool to assume free format for the input file.
- fixed**
 - Forces the tool to assume fixed format for the input file.
- I pathname**
 - Add pathname to the list of directories which are to be searched for module information (.mod) files and include files. The current working directory is always searched first, then any directories named in -I options.
- mdir dir**
 - Write any module information (.mod) files to the directory *dir* instead of the current working directory.
- dusty**
 - Allows the compilation and execution of ``legacy'' software by downgrading the category of common errors found in such software from ``Error'' to ``Warning'' (which may then be suppressed entirely with the -w option).
- strict95**
 - Produce obsolescence warning messages for use of ``CHARACTER*'' syntax. This message is not produced by default since most programs contain this syntax.

nag_modules95 notes

❑ nag_modules95:

- ❑ contains the Fortran 95 parser
- ❑ requires access to all USEd modules

❑ For the routine VECADD.f90

- ❑ USEs modules GLOBALS90 and UTILS90 – modules needing processing with nag_modules95 as well.
- ❑ *nag_modules95 -free -strict95 -I../globals90 -I../utils90 vecadd.f90*

❑ Processing a complete program:

- ❑ USEs module FELIB90 (which uses many others)
- ❑ *nag_modules95 -free -strict95 -I../modules90 seg3p1.f90*
 - this will flag that source codes were not available
 - they will need adding to the -I list

The NagWare f95 parser

- ❑ N.B. The NagWare f95 Parser, which is part of nag_cbm95, nag_decs95, nag_prest95, nag_fcalls95, nag_ustan95 and nag_mkintf95, requires to read pre-compiled module files (``.mod" files) for any modules that are USEd but are not previously defined in the current file.
- ❑ If you have compiled your Fortran 95 source code with the NAGWare f95 Compiler, the ``.mod" files that were created by the compiler are those required by the Tools. If you only want to create the ``.mod" files for the Fortran 95 source file example.f90, the simplest command is;
`f95 -M example.f90`
- ❑ The ``-M" option will stop the compiler after module files have been output. Any successful compile command will create these ``.mod" files, or the directory specified by *-mdir*.

nag_pfort - NAGWare f77 Tools Portability Verifier

- ❑ **nag_pfort is the NAGWare f77 Tools portability verifier.**

nag_pfort [options] ... *file* ...

- ❑ **nag_pfort checks for:**

- ❑ conformance to the NAGWare f77 Tools language standard
- ❑ conformance to the ANSI Fortran 77 standard,
- ❑ conformance to a portable subset of the ANSI Fortran 77 standard correct inter-program-unit communication, and
- ❑ unsafe references, which are defined below.

- ❑ **Some of the options..**

- ❑ **-obs90** checks for features listed as Obsolescent in Fortran 90
- ❑ **-obs95** checks for features listed as Obsolescent in the Fortran 95 standard
- ❑ **-dep** checks for a number of deprecated features
- ❑ **-port90** allows some Fortran 90 features
- ❑ **..rather many more**