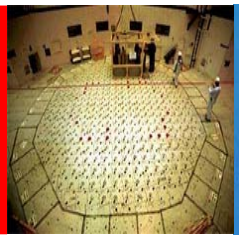
MALPAS

Advanced Software Analysis and Verification

Introductory Guide



Issue 2

Atkins
The Barbican, East Street
Farnham, Surrey, UK

Visit our web-site at www.atkinsglobal.com

Contents

This page is left intentionally blank

Introduction

High Integrity Software

As systems become increasingly complex, many designers are turning to software to provide the facilities they need. Software can provide complex functionality at low power, low weight and high reliability. Unfortunately, realising all this potential has proved difficult. In addition to simple coding mistakes, software is highly prone to systematic “designed-in” errors. For the majority of applications, errors may not matter unduly, so long as they rarely cause failures; but in safety critical applications, any failure can have dire consequences. To obtain confidence in the software specification and its implementation, a process of rigorous verification is needed.

The Role of MALPAS

MALPAS is one of the most rigorous verification tools on the market today. By subjecting software to comprehensive analysis, MALPAS will reveal program structure, examine functionality in detail, and check conformance with the specification. Users throughout the world (Germany, Canada, UK, France, Australia) in sectors varying from security, aerospace, defence, nuclear power and transport have employed MALPAS to:

- Improve the integrity of their systems
- Demonstrate to regulatory bodies and customers that their programs are error free
- Create critical software more cost effectively, i.e. delivered on time and within budget

Malpas Approach

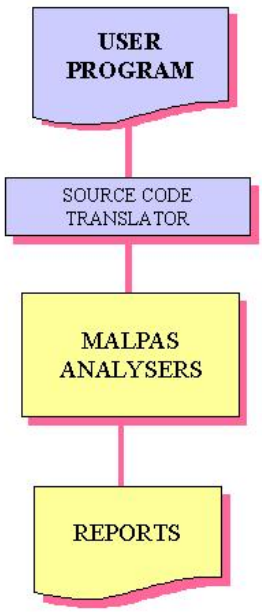
The MALPAS approach to static analysis is being used increasingly in critical software development and verification. It has important advantages compared with other verification strategies:

- **Coverage** - all possible paths through the program are analysed, not just those foreseen by the programmer
- **Cost-effectiveness** - MALPAS does not require the production of expensive test-rigs or the collection of test datasets. Because any individual module of code can be analysed as soon as it is completed, any quality problems will be uncovered early in the lifecycle
- **Versatility** - MALPAS remains effective throughout the whole lifecycle, from specification through to in-service maintenance. MALPAS is also available for a number of computing platforms from VAX to Windows NT

Aim of this Guide

This guide aims to:

- give project managers and software engineers information on the main features of MALPAS
- outline the practical benefits of using MALPAS
- indicate how MALPAS can be acquired and supported



The MALPAS Analysis Process

MALPAS supports a hierarchy of analysis techniques of increasing rigour, from structural analyses to formal proof.

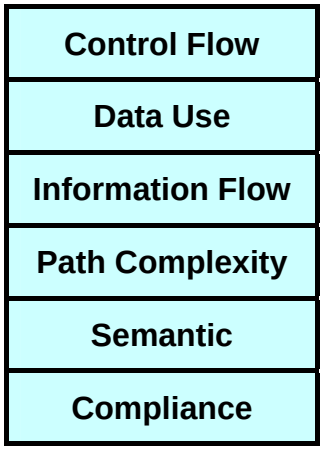
Rest of the Document

- Chapter 2 gives a brief overview of this hierarchy.
- Chapter 3 discusses the practical usage of MALPAS and its effects on software certification productivity, and maintenance.
- Chapter 4 describes the MALPAS licensing and training services provided by Atkins.
- For those interested in the details of the technical capabilities of MALPAS, a short but realistic example of MALPAS analysis is provided in a companion document, "MALPAS – Advanced Software Analysis and Verification – Example Analysis".

MALPAS

MALPAS Capability

MALPAS supports a hierarchy of six static analysis techniques, each of which investigates a different aspect of programs, and may be applied individually or sequentially. Each successive technique in the MALPAS hierarchy is designed to yield increasingly rigorous insights into the structure and logic of a program.



The MALPAS Hierarchy of Analysers

Analysis Techniques

Control Flow Analysis - identifies unstructured control flow in a program, for example, multiple entry and exit points, unreachable code, and infinitely looping dynamic halts. It gives an initial feel for the structure of the program and is particularly useful for assessing assembly code that lacks the high-level language control structures.

Data Use Analysis - identifies all the inputs and outputs of each procedure whether documented or not, including any surreptitious use of global data. Data handling errors are also revealed: for example, there are checks that each variable is initialised before being read.

Information Flow Analysis - each procedure is analysed to deduce the information on which it outputs depend. The analysis of these dependencies is the first step toward

identifying serious errors in the dynamic behaviour of the program, such as unwanted side-effects.

Path Assessor - the number of paths through the code is reported for each procedure. This complexity measure has a direct bearing on the difficulty, and therefore the cost, of testing the procedure.

Semantic Analysis - this analysis re-expresses the complete semantics of the procedure in a formal mathematical notation that is more explicit than the code. The analyst can use this algebraic representation to check for discrepancies between the program's behaviour and its specification.

Compliance Analysis - supports a formal proof that a procedure behaves in accordance with its specification; in other words, that if the required precondition holds when the procedure is called, the required postcondition must hold when it terminates. Where this is not the case, compliance analysis will identify the input values for which the procedure will fail.

IL – MALPAS Intermediate Language

MALPAS is used on software source code, but prior to analysis it is necessary to model the source code in the MALPAS Intermediate Language (IL). Given a translation of the code into the IL, MALPAS can be applied programs of any programming language. The construction of the IL model is a highly instructive exercise in its own right; users have found that re-expressing a program using the formal rigour of the IL quickly reveals ambiguities in the code. Errors and potential improvements often become apparent even at this early stage. The translation of the source code into IL may be performed either by hand or by use of an automatic translator. Several automatic translators are available for a variety of languages including Ada, C, Pascal, CORAL 66, PL/M-86, FORTRAN, Intel ASM86 and Motorola 6809.

Applications Assessed by MALPAS

MALPAS has been applied to safety, security, mission, and business critical applications. Some of these are listed below:

- The Temelin Nuclear Power Station - Primary & Diverse Reactor Protection Systems
- The Sizewell B Nuclear Power Station - Primary Protection System (see Refs. 7 & 10)
- Lockheed Martin C130J (various safety critical LRUs)
- The BR710 Digital Engine Control Unit
- Inlet Guide Vane Controller for Pegasus engine (for the Harrier jump-jet)
- The Swordfish Torpedo Arming System
- A Smart Timer Fuse Program
- An Ammunition Storage Control System
- The BAE Systems Tornado Auto Wing Sweep System

- The Merlin Helicopter Stores Management System
- The Astra Hawk Fly-by-Wire Control System
- The Dungeness B Single Channel Trip System
- An Airborne Collision Avoidance System
- A Flight Vehicle Self-Destruct System
- The Australian Railway Signalling System
- A Gas Distribution Network Control System
- Secure Software for Government and Armed Services Communications
- A Medical Infusion Pump

In practice, non-critical applications will benefit from MALPAS analysis.

Using MALPAS

The versatility of MALPAS has been proven through more than a decade of development and use throughout the software development life cycle, in a broad range of sectors.

Specifications in MALPAS IL

The MALPAS Intermediate Language (IL) is a powerful notation that can be used to specify any complex problem in a rigorous way. The specification can be written in any of the common specification styles; algebraic, functional, or model-based.

Source Code Verification

The normal assessment methods for source code fall into two distinct categories: Dynamic Analysis (e.g. animation, testing) and Static Analysis. Dynamic Analysis works by exercising the code over a set of test conditions. It is based on engineering judgement: test conditions have to be chosen to maximise the likelihood that if the code behaves correctly for the test conditions, it will work when it encounters all other conditions. Static Analysis is based on logic: verification is by logical argument rather than execution. It can be applied with varying degrees of rigour, from code walkthroughs and inspections through to fully formal proof. MALPAS can be used to support both dynamic and static analyses, to any level of rigour, but it is most closely associated with Static Analysis. A realistic illustration of its capabilities is given in the companion document, "MALPAS – Advanced Software Analysis and Verification – Example Analysis".

In practice, dynamic and static analyses are complementary: testing can be used to check the assumptions in logical arguments and logical arguments can be used to check deficiencies in testing.

Dynamic Analysis provides confidence that the code works reliably in foreseeable situations. It establishes that the software and hardware work together as planned, and provides some assurance that a compiler has not introduced unwanted errors. However, latent errors may emerge only on rare occasions and may be very difficult to trace. Furthermore, latent errors may be activated as a side-effect of maintenance.

Static Analysis is a significant aid to assessments of the technical difficulties that could emerge when code has to be modified. It is therefore a sound indicator of future lifecycle costs.

Cost Effective Process

MALPAS supports cost-effective Static Analysis by:

- breaking down the assessment process into definable subtasks, which aids in formalising and planning the assessment task
- providing automated support for the consideration of the critical code for *all* input conditions and *all* execution paths, and not just those that have been tested
- focusing attention on the parts of the code most likely to contain errors

Dynamic Analysis Support

A well-focused Dynamic Analysis requires careful consideration of the code under test. MALPAS supports dynamic analysis by:

- identifying the inputs upon which critical behaviour depends
- identifying every feasible path through the code in order to assess the extent of test coverage that can be achieved
- permitting animation of the specification to find the expected outputs

Error Discovery

Practical experience shows that a great variety of software errors can be found by using MALPAS, from simple coding errors, through to unforeseen pathological behaviours and faulty error-handling routines. Many of the examples in the following list were found in software that was previously thought to have been adequately tested:

- incorrect choice of logical operators: e.g. the use of “less than” instead of “less than or equal to”
- uncommon but pathological inputs: e.g. a 32 bit averaging routine was found to fail if both inputs were 8000001_{16}
- unguarded paths through the code that would be hazardous if executed
- corruption of registers across assembly language procedure calls
- dynamic type errors: e.g. confusion a pointer-to-byte value with pointer-to-word value in an assembler
- built-in-test failures: e.g. failure of a RAM testing algorithm for particular sizes of memory

- misinformation: incorrect error reporting on an operator maintenance panel
- faulty configuration of hardware: erroneous mixing of small and large memory models for the 80x86 processor
- generation of false alarms in safety monitoring systems

Certification

MALPAS is widely recognised as providing the full range of static analysis techniques mandated in current standards such as Def Stan 00-55 (Ref. 4), IEC 61508 (ref. 5) and DO-178B (Ref. 6). In addition, it has become a de-facto standard for some assessment and regulatory bodies. For example, analysis of military safety critical airborne systems is encouraged up to Semantic Analysis level. The nuclear industry prefers analysis right through to Compliance Analysis for safety critical code. The security industry defines the levels of Static Analysis required for different levels of security in terms that map directly onto the MALPAS analysis hierarchy.

Maintenance

If the documentation is of such poor quality that it cannot be used to verify the code, an alternative use of MALPAS is to reverse engineer proper documentation. MALPAS can rapidly build up a complete picture of the behaviour of a piece of code, which is then used to drive the code documentation. Engineers can inspect MALPAS reports to decide if the behaviour of the code is sensible within the context of the application. Such documentation will identify all interactions within the code, and so make it significantly easier to maintain.

Other Applications

As experience with the tool has grown, novel applications for MALPAS have been found. Two such applications are described here.

In a fly-by-wire aircraft control system, it was necessary to ensure that certain tasks were completed within a certain time. MALPAS was used to calculate the worst case run time for these tasks. Firstly, the assembly code was translated into MALPAS IL in such a way as to highlight the amount of time taken by each instruction. Secondly, the IL was submitted for Semantic Analysis that identified the time for each path through the program. The worst-case time could readily be identified.

In a critical nuclear power application, concern had been expressed about the possibility of errors introduced by the compiler used for a safety critical application. Typically, compilers are not developed to safety critical standards, and so it was decided that the object code must be independently verified against the source code. This was achieved by translating both the source code and the object code into IL and then comparing the programs using MALPAS Compliance Analysis. The whole process could be highly automated since

both the source and object languages were rigorously defined (Ref. 9).

Productivity

The effort required to use MALPAS depends on several factors such as analyst experience, choice of programming language and style, the standard of documentation, and the amount of code to be analysed, and the depth to which the MALPAS analysis hierarchy is to be applied. However, the table below can be used as a rule-of-thumb estimate of analysis time. As experience is gained on a particular application, more accurate estimates of the time taken can be calculated specific to the application.

Analysis technique	Source lines per day
Control Flow Analysis, Data Use Analysis and Information Flow Analysis	100-500
Semantic Analysis	20-100
Compliance Analysis	10-50

Conclusions

MALPAS is a powerful tool for assuring that code is fit for purpose and correct. It can be used to support many parts of the software development process and has been proven across a wide variety of applications. Many supplier organizations, assessment laboratories, and regulatory bodies, having recognised the value of static analysis, now insist on its use for safety and security critical code. The tool is fully supported by Atkins, who also offer an independent assessment service.

Acquiring MALPAS

Licensing

Worldwide marketing rights for MALPAS are held by Atkins.

The MALPAS licensing system is very flexible. Typically, project specific licenses are granted, but the tool can also be leased. Evaluation packages are also available which include the training, support, and documentation needed to become familiar with the toolset.

Compatibility

MALPAS is compatible with all systems running Microsoft Windows 9x/NT/2000 and XP.

MALPAS Translators

The MALPAS product range includes automatic translators for Ada83, C, Pascal, FORTRAN, Intel PL/M86, CORAL66, ASM86 and Motorola M6809. If a suitable automatic translator is not available, then two options are possible.

Hand Translation: For moderate amounts of code, it is possible to translate the source code into IL by hand. The flexibility of IL makes this a reasonable approach, particularly for assembler code. In the past, such hand translation has been used for M68020 and Z80 code.

Bespoke Translator: We have produced several bespoke translators at the request of users. Alternatively, it is possible to customise an existing automatic translator in the light of project-specific programming practices.

Training

We offer a range of courses specific to the MALPAS tool, and also more wide ranging courses on static analysis and software verification. These include:

- Software Static Analysis (2 days)
- MALPAS Introductory Course (3 days)
- Compliance Analysis Workshop (3 days)

The 3 day introductory course aims to:

- Illustrate MALPAS's potential as a verification tool to improve software productivity and enhance integrity
- Give delegates extensive hands-on experience with the tool so they can use it with confidence on their own applications
- Demonstrate how MALPAS handles many of the major computing languages in use today e.g. Ada, C, Pascal and Assembler

MALPAS Support Service

In addition to developing the tool and presenting a range of training courses, Atkins provides support services on payment of an annual support fee:

- the annual MALPAS User Group run by the users with administrative support from Atkins
- on-line telephone support and advice covering, installation and general trouble shooting
- seminars on the background and theory behind MALPAS and Static Analysis

The optional support service includes product maintenance with regular program updates supplied to users at no additional charge.

Software Assessment Service

Finally, rather than climbing the learning curve, several customers prefer to leave the work to Atkins. We have a team of experienced analysts with a broad knowledge of software engineering who can offer a fully independent software assessment service. This can range from an assessment of the software development process, to a detailed MALPAS analysis of the source code itself.

For more information on any of the above, please contact:

Chris Sampson – Software Assessment Consultant
chris.sampson@atkinsglobal.com

Andy Hodgkinson – MALPAS Products Manager
andy.hodgkinson@atkinsglobal.com