

First edition
2000-11-01

Road vehicles — Development guidelines for vehicle based software

*Véhicules routiers — Guide pour le développement de logiciels installés à
bord de véhicules*



Reference number
ISO/TR 15497:2000(E)

AVA **Documents**
© ISO 2000
Document Sharing Hub

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO 2000

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

In exceptional circumstances, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide by a simple majority vote of its participating members to publish a Technical Report. A Technical Report is entirely informative in nature and does not have to be reviewed until the data it provides are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this Technical Report may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/TR 15497 was prepared by the United Kingdom Motor Industry Software Reliability Association (MISRA) as guidelines published in 1994, and was adopted (without modifications except those stated in clause 2 of this International Standard) by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

Road vehicles — Development guidelines for vehicle based software

1 Scope

This Technical Report provides safety-related guidelines for the development of vehicle based software.

2 Recommendations

The technical recommendations are those made in the following publication (reproduced on the following pages), which is adopted as a Technical Report:

Development Guidelines for Vehicle Based Software, Motor Industry Software Reliability Association (MISRA), United Kingdom, 1994.

For the purposes of international standardization, the modifications outlined below shall apply to the specific clause and paragraphs of the MISRA publication.

Page i to iv (of the MISRA publication)

This is information relevant to the MISRA publication only.

Page 73-75

Clause 6

Substitute the following for the corresponding references.

[10] ISO 9001, *Quality systems — Model for quality assurance in design, development, production, installation and servicing*.

[15] ISO 11748 (all parts) —¹⁾, *Road vehicles — Technical documentation of electrical and electronic systems*.

[25] ISO 11898, *Road vehicles — Interchange of digital information — Controller area network (CAN) for high-speed communication*.

[26] ISO 11519 (all parts), *Road vehicles — Low-speed serial data communication*.

[29] ISO 14230 (all parts), *Road vehicles — Diagnostic systems — Keyword Protocol 2000*.

Insert the following reference.

[24] EMC standards and technical reports applicable to road vehicles:

ISO 7637 (all parts), *Road vehicles — Electrical disturbance by conduction and coupling*;

ISO /TR 10305, *Road vehicles — Generation of standard EM field for calibration of power density meters from 20 kHz to 1 000 MHz*;

ISO/TR 10605, *Road vehicles — Electrical disturbances from electrostatic discharge*;

1) To be published.

ISO 11451 (all parts), *Road vehicles — Vehicle test methods for electrical disturbances by narrowband radiated electromagnetic energy*;

ISO 11452 (all parts), *Road vehicles — Component test methods for electrical disturbances by narrowband radiated electromagnetic energy*.

3 Revision of the MISRA publication

It has been agreed with the Motor Industry Software Reliability Association that ISO/TC 22/SC 3 will be consulted in the event of any revision or amendment of the MISRA publication. To this end, the British Standards Institution (BSI) will act as a liaison body between MISRA and ISO.

First published November 1994
by The Motor Industry Research Association
Watling Street
Nuneaton
Warwickshire
CV10 0TU

<http://www.misra.org.uk>

© The Motor Industry Research Association, 1994, 2000

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical or photocopying, recording or otherwise without the prior written permission of the Publisher.

ISBN 0 9524156 0 7

British Library Cataloguing in Publication Data.

A catalogue record for this book is available from the British Library.





The Motor Industry Software Reliability Association

Development Guidelines For Vehicle Based Software

November 1994

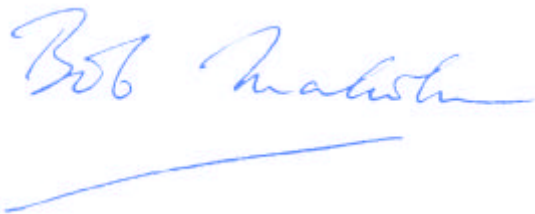
Foreword

As coordinator of the Safety Critical Systems Research Programme, supported by the Department of Trade and Industry and the Engineering and Physical Sciences Research Council, I am pleased to support the publication of these guidelines.

In this programme we have been concerned to ensure that the results of research should not be buried in learned papers, but promulgated in ways which will affect practice in industry. We have therefore sought the involvement of user organizations so that the work based on an understanding of their real industrial needs, and so that the results will be credible to their peers. The MISRA Study, with its combination of vehicle and equipment manufacturers, has admirably realized that ambition.

The voluntary nature of the guidelines is important. They were produced voluntarily, for the benefit of both the industry and the public. Their adoption will also be voluntary. This has encouraged the MISRA consortium to develop guidelines which offer genuine benefits for their users, rather than burdensome restrictions.

The renowned cost-consciousness of the automotive industry might be thought by some to diminish its contribution to technological development. On the contrary: while this industry is properly cautious in the interests of safety, the fiercely competitive market and the public exposure of any problems ensures that the commercial value of new technology is thoroughly evaluated. This sector is a hard proving-ground for technology. So, while many of the issues on which the MISRA consortium has concentrated are specific to vehicles, other sectors might do well to see what they might glean from the trends here.



Bob Malcolm
DTI-EPSRC Safety Critical systems



Foreword (continued)

The motorist gains many benefits, including enhanced safety features, from advances in vehicle electronics.

The development of software is a specialized and often complex area where much relies on an effective approach by those directly involved. I welcome MISRA's initiative and efforts in developing these safety related guidelines and advice under a joint DTI, EPSRC and industry funded programme. The guidelines reflect a responsible and serious industry attitude to safety issues.

Malcolm Fendick BSc CEng FIMechE
Chief Mechanical Engineer
Department of Transport

The use of electronic systems in vehicles has increased significantly over recent years and will continue to increase. Much time and resources are being committed by vehicle manufacturers to deal with the compatibility of such systems in particular the problem of interference from external sources.

The vehicle owner expects his vehicle to be reliable and safe, and this includes the electronic systems. Electronic systems are dependent on the software provided by the manufacturer. The greater complexity of such systems is increasing the need to maintain software quality and reliability. This has resulted in the need for a unified approach to software design. It is therefore pleasing that an independent group has produced these guidelines on vehicle-based software which will be of benefit to the Motor Industry and also more importantly the motorist.

K B Barnes
Head of Engineering
SMMT



Acknowledgements

The MISRA consortium would like to thank the following organizations for their support of the study that produced these Guidelines:

AB Automotive Electronics Ltd	(CM)
AP Borg & Beck	(CM)
Delco Electronics	(CM)
Department of Trade and Industry	
Department of Transport	
Ford Motor Company Ltd	(CM)
Jaguar Cars Ltd	(CM)
Lotus Engineering	(CM)
Lucas Electronics	(CM)
Rolls-Royce and Associates Ltd	(C)
Rover Group Ltd	(CM)
The Centre for Software Engineering Ltd	(C)
The Motor Industry Research Association	(PM, C)
The Society of Motor Manufacturers and Traders Ltd	
The University of Leeds	(C)

Key

CM	controlling member of the MISRA consortium
PM	project manager
C	consultant

The MISRA consortium would like to thank the following individuals for their contributions to these Guidelines:

Alex Abbot	Vivien Hamilton	Frank O'Neill
Klaus Allen	Nick Heatley	Dave Perry
Neil Andrewartha	Keith Hobley	Mike Radford
Nigel Barrett	Peter Jesty	Roger Rivett
Terry Beadman	Edward Jones	Steve de la Salle
Nick Bennett	Ian Kendall	Chris Shakespeare
Colin Bowles	Keith Longmore	Heather Storey
Tom Buckley	Paul Manford	Ian Stothers
John Comfort	Chris Marshall	Lloyd Thomas
Nick Dunn	Josh McCallin	David Ward
Paul Edwards	Tom Monk	Brian Whitfield
Simon Farrall	Tony Moon	Anthony Wood
John Fox	Dave Newman	Marco Zasiura



Contents

	Page
1. Introduction	1
1.1 Statement of mission and objectives.....	1
1.2 Benefits to the end customer	1
1.3 The MISRA consortium	1
1.4 Background	2
1.5 Scope and uses of the Guidelines	3
1.5.1 Scope	3
1.5.2 Uses	3
1.6 Fundamental concepts	5
2. Definition of terms	7
2.1 Definitions	7
2.2 List of abbreviations	7
3. Software lifecycle	8
3.1 Project planning	8
3.1.1 Project definition	8
3.1.2 Lifecycle plans	9
3.1.3 Planning for verification and validation	9
3.1.4 Assessment	13
3.1.5 Reuse	14
3.2 Integrity	14
3.2.1 Introduction	14
3.2.2 Safety analysis	15
3.2.3 Human factors in safety analysis	18
3.2.4 Development approaches	19
3.3 Requirements specification	22
3.3.1 Whole vehicle architecture	22
3.3.2 Vehicle control systems	25
3.3.3 Noise and electromagnetic compatibility	28
3.3.4 Verification and validation of software requirements	30
3.3.5 Tools and techniques for requirements specification	32
3.4 Design	33
3.4.1 Real-time implications	33
3.4.2 Floating point arithmetic	36
3.4.3 Modelling	37
3.4.4 Optimization and adaptive control	38
3.4.5 Communications and multiplexing	38
3.4.6 On-board diagnostics	41
3.4.7 System security	43
3.4.8 Fault management	43
3.4.9 Design for verification and validation	45
3.4.10 Tools and techniques for design	46



Contents (continued)

3.5	Programming	47
3.5.1	Codes of practice	47
3.5.2	Verification and validation of code	48
3.5.3	Programming tools and techniques	48
3.6	Testing	49
3.6.1	General	49
3.6.2	Dynamic test	49
3.6.3	Integration test	49
3.6.4	System test	51
3.6.5	Tools and techniques for testing	51
3.7	Product support	52
3.7.1	Off-board diagnostics	52
3.7.2	Software maintenance	53
4.	Software quality planning	55
4.1	Management responsibilities	55
4.2	Education and experience	56
4.3	Human factors in software development	56
4.3.1	Introduction	56
4.3.2	Teams and organizational structure	57
4.3.3	Individual differences and job design	57
4.3.4	Human error management	58
4.3.5	The physical environment	58
4.4	Quality assurance	59
4.4.1	Standards and accreditation	59
4.4.2	Checklists	59
4.4.3	Assessment of compliance	59
4.4.4	Changes during production	60
4.4.5	Software process metrics	60
4.5	Documentation requirements	62
4.6	Subcontracting	63
4.6.1	Introduction	63
4.6.2	Definitions	63
4.6.3	Technical considerations	65
4.6.4	Commercial considerations	67
5.	Emerging technologies	70
5.1	General	70
5.2	Neural networks	70
5.3	Object orientation	71
5.4	Fuzzy logic	71
5.5	Formal mathematical methods	72
6.	References	73
7.	Index	76



1. Introduction

1.1 Statement of mission and objectives

1.1.1 The purpose of these Guidelines is to provide assistance to the automotive industry in the creation and application within a vehicle system of safe, reliable software.

1.1.2 There has been much recent growth in the quantity and complexity of electronic controls on motor vehicles. The greater the complexity, the harder it is to maintain the software quality and reliability the customer has come to expect. A unified approach to software development, using agreed techniques, is desirable across the automotive industry. This will enable the driver or owner to have continued confidence in complex electronic systems.

1.2 Benefits to the end customer

1.2.1 The public exposure to software in applications with safety implications is increasing. It may be through vehicles that the majority of the public will encounter such software. Therefore, it is vital that such software is both correct and perceived to be correct.

1.2.2 The record of the automotive industry in regard to software is good. The application of these Guidelines will maintain the industry's confidence in the quality of the software used in its products as the complexity increases.

1.3 The MISRA consortium

1.3.1 The MISRA consortium was formed in response to the UK Safety Critical Systems Research Programme, supported by the Department of Trade and Industry and the Engineering and Physical Sciences Research Council, and consisted of eight controlling members, four consultants and a project manager (see Acknowledgements page).

1.3.2 The consortium initially conducted a survey of existing work, both in the automotive sector and in other industrial sectors.

1.3.3 The consortium then formed eight subgroups to research specific issues relating to automotive software:

- diagnostics and integrated vehicle systems
- integrity
- noise, EMC and real-time
- software in control systems
- software metrics
- verification and validation
- subcontracting of automotive software
- human factors in software development.



Introduction (continued)

- 1.3.4 Each group produced a report [1]–[8] that provides more detailed supporting information, and recommendations that have been incorporated into these Guidelines.
- 1.3.5 The results of the survey were continually updated throughout the study [9].
- 1.3.6 The eight reports and the survey report are available separately [1]–[9]. These reports provide additional background, more detailed recommendations and additional references that will be invaluable to the specialists in the field.
- 1.3.7 For the sake of brevity, the Guidelines do not always justify recommendations. The eight reports [1]–[8] contain the background material and justifications where appropriate.

1.4 Background

- 1.4.1 There are important differences between software and other forms of automotive engineering and components:
- (a) Software is primarily a design, with no manufacturing variation, wear, corrosion or ageing aspects.
 - (b) It has a much greater capacity to contain complexity.
 - (c) It is perceived to be easy to change.
 - (d) Software errors are systematic, not random.
 - (e) It is intangible.
- 1.4.2 There are differences between automotive applications and applications in other industrial sectors:
- (a) Production volumes are high (leading to manufacturing variation), related mechanical components are subject to wear and maintenance levels are difficult to assure. Therefore automotive software has an emphasis on data driven algorithms, parameter optimization, adaptive control and on-board diagnostics.
 - (b) Passenger car drivers receive little or no training compared with other users of computer-based products and services. Therefore automotive software requires an emphasis on failure management techniques based on the controllability of the vehicle.
 - (c) Traditional automotive test environments use real vehicles and components, as well as simulations. These are available to test systems and software extensively and safely before they reach the customer.



Introduction (continued)

1.5 Scope and uses of the Guidelines

1.5.1 Scope

- 1.5.1.1** The boundaries between software and system design are not precise and there are many interrelated aspects. These Guidelines take a broad view of software, recognizing that many concerns attributed to software reliability are as much systems issues as they are related to software engineering.
- 1.5.1.2** The Guidelines make recommendations on system issues that are considered to influence software development.
- 1.5.1.3** The Guidelines are forward looking, supporting both today's developments and the vehicle features that are envisaged over the next decade.
- 1.5.1.4** The discipline of software engineering is already well supported by academia, text books and standards. Further guidance in this field is not appropriate at this time.
- 1.5.1.5** The eight specific issues for which guidance is given can be seen as aspects that provide a link between the established automotive engineering disciplines and software engineering, as shown in Figure 1.

1.5.2 Uses

- 1.5.2.1** This document is intended to be used by software engineers, managers and others involved in the creation, procurement and maintenance of embedded vehicle software.
- 1.5.2.2** The following uses are intended:
- guidance for creating contracts and specifications for software procurement
 - as an introduction to issues of automotive software reliability
 - the basis for training requirements within the automotive industry
 - guidance for company quality procedures
 - guidance for management on resource requirements
 - to provide the basis of assessment
 - as a foundation for a standard.



Introduction (continued)

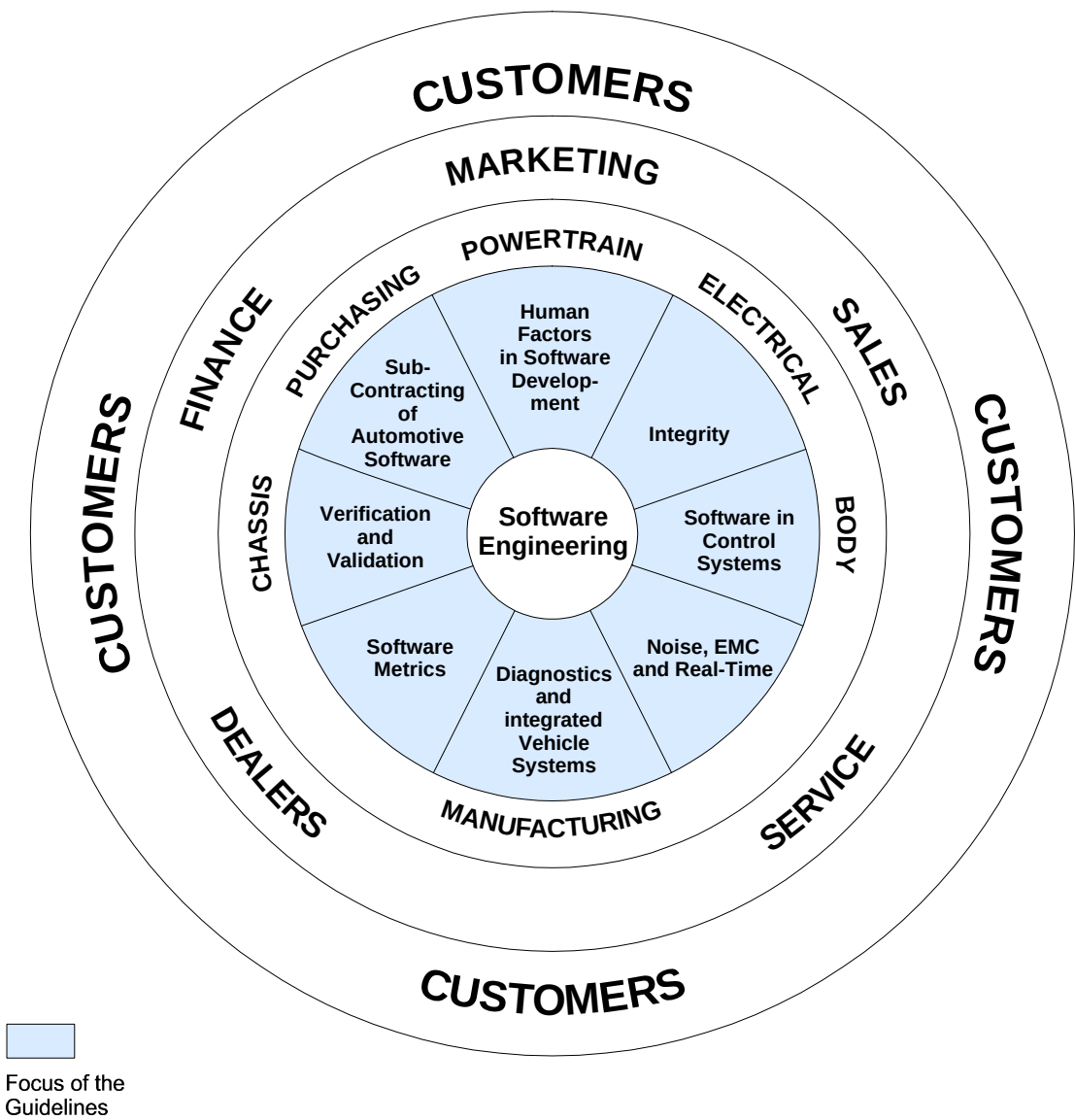


Figure 1. Scope of the MISRA Guidelines



Introduction (continued)

- 1.5.2.3 Many of the recommendations are interrelated and overlap. In some instances, due to their generic nature, they may appear not to be in the optimum order within the document.
- 1.5.2.4 Not all the recommendations will apply in any one situation, so it is not anticipated that any project will, or can, conform to all the recommendations. Reasons for deviation from relevant parts of the Guidelines should be properly justified in the associated documentation. The Guidelines make recommendations on assessment of compliance.

1.6 Fundamental concepts

- 1.6.1 It is assumed and recommended that the users of this document operate a Quality Management System capable of meeting the requirements of ISO 9001 [10] as assessed under the guidelines in ISO 9000-3 [11] (e.g. TickIT [12]).
- 1.6.2 The users should have a working knowledge of the relevant software engineering practices.
- 1.6.3 Users of these Guidelines should be aware of present and emerging standards. Standards of particular importance include:
- the emerging generic standards in preparation by the International Electrotechnical Commission (IEC) [13, 14]
 - the work of the ISO/TC22/SC3/WG11, Automotive Electronic Control Systems — Technical Documentation, working party [15]
 - the IEEE software engineering standards collection [16]
 - the IEE guidelines for the documentation of computer software [17].
- 1.6.4 Compliance with these Guidelines or with any standard does not of itself confer immunity from legal obligations.
- 1.6.5 These Guidelines are not intended to be prescriptive or proscriptive. As an element of flexibility it is important to allow users to adopt an approach that best matches their processes and organizational strengths. Hence the term “should” is normally used throughout in preference to “must” or “shall”. No particular significance should be attached to the use of alternative wording.
- 1.6.6 Techniques, activities and their rigour increase with integrity level. Equally, the weight of the relevant recommendations increases.
- 1.6.7 For the sake of readability there are no cross-references in this document.



Introduction (continued)

- 1.6.8 Several basic principles associated with safety engineering and safety related systems design need to be understood:
- (a) Safety, like justice and democracy, must be seen to be present.
 - (b) The greater the risk, the greater the need for information.
 - (c) Software robustness, reliability and safety, like quality, should be built in rather than added on.
 - (d) The requirements for human safety and security of property can be in conflict. Safety must take precedence.
 - (e) System design should consider both random and systematic faults.
 - (f) It is necessary to demonstrate robustness, not rely on the absence of failures.
 - (g) Safety considerations should apply across the design, manufacture, operation, servicing and disposal of products.



2. Definition of terms

2.1 Definitions

2.1.1 The terminology used in these Guidelines is taken from [13, 14, 18].

2.1.2 Terms specific to the automotive industry are defined in the text.

2.2 List of abbreviations

CAN	controller area network
CARB	California Air Resources Board (US)
CASE	computer aided software engineering
CV	<i>curriculum vitae</i>
DIS	draft international standard
EC	European Commission
ECU	electronic control unit
EMC	electromagnetic compatibility
EMI	electromagnetic interference
IEC	International Electrotechnical Commission
IEE	The Institution of Electrical Engineers
IEEE	The Institute of Electrical and Electronics Engineers (US)
I/O	input and output
ISO	International Organization for Standardization
KWP	keyword protocol
OBD	on-board diagnostics
OSI	open systems interconnection
QA	quality assurance
RAM	random-access memory
ROM	read-only memory
SAE	Society of Automotive Engineers (US)
VAN	vehicle area network
VDU	visual display unit



3. Software lifecycle

3.1 Project planning

3.1.1 Project definition

- 3.1.1.1** The use of software has many benefits in terms of cost, flexibility and functionality. However, because of the problems associated with complexity, it is important that systems containing software are only used where these benefits are required.
- 3.1.1.2** The recommendations in this section can be applied both at a vehicle system level and at a software system level [2, 6].
- 3.1.1.3** The project objectives should be clearly laid down before the start of a project. For example, the nature of a project will be different for a volume production system when compared to a research and development prototype.
- 3.1.1.4** The project definition should consist of a list of features and functions to be implemented. It should be agreed and documented by the appropriate authorities of the vehicle manufacturer and made available to the design and development teams before detailed work begins.
- 3.1.1.5** The project definition should also include any known legislative requirements, project assumptions and nonfunctional requirements (e.g. the use of an existing engine and/or ECU in a new vehicle).
- 3.1.1.6** It is particularly important to lay down and agree which features are fixed and which are customer options.
- 3.1.1.7** The project definition should be subject to strict change control.
- 3.1.1.8** Any changes made to the project definition will lead to increased risk from both safety and commercial points of view. The later changes are made, the greater the incremental risk.
- 3.1.1.9** Computer-based tools to be used during the project should be identified, and provision made for their procurement and resourcing at the project definition stage. Such tools are likely to include:
- configuration management tools
 - calibration development aids
 - modelling and simulation tools
 - CASE tools
 - rapid prototyping tools.



Software lifecycle (continued)

3.1.2 Lifecycle plans

3.1.2.1 An appropriate development approach should be defined and documented (e.g. by producing a Software Development Plan, a Quality Plan, a Safety Plan and a Configuration Management Plan). The design material should define:

- (a) project organization.
- (b) procedures to be followed.
- (c) management processes to ensure compliance with the procedures in (b).
- (d) planned management reviews of the effectiveness of the procedures in (b) and processes in (c).
- (e) specific software deliverables together with their authors and readers.

3.1.2.2 The project team should progressively produce a Software Development Plan with clearly defined phases that, comprehensively and cost-effectively, cover the:

- specification
- design
- programming
- integration
- verification and validation (including testing)
- in-service support of the software.

3.1.2.3 The Safety Plan [14] should aim to ensure that verification and validation are performed throughout the project. These activities should be performed with a rigour appropriate to the integrity level.

3.1.2.4 Appropriate lifecycle plans should be defined before the start of a project for the system and the software. Figure 2 shows an example lifecycle.

3.1.2.5 Each individual system should have its own lifecycle that should be compatible with the project definition and the overall vehicle plan. An example is shown in Figure 3.

3.1.3 Planning for verification and validation

3.1.3.1 The verification and validation activities should be defined to establish a level of confidence in the software that corresponds the integrity requirements of the system. See Figure 4.



Software lifecycle (continued)

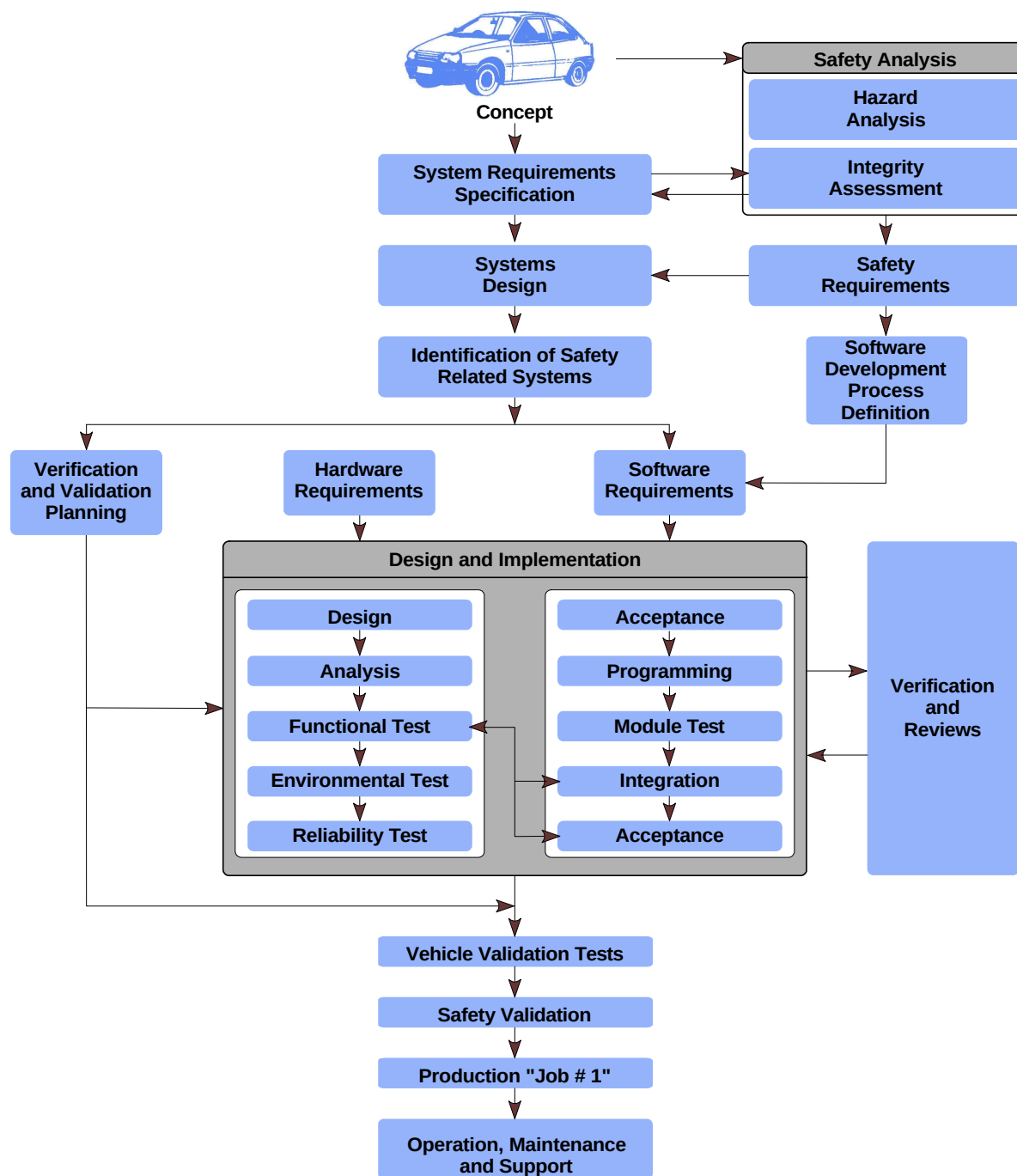


Figure 2. An example lifecycle



Software lifecycle (continued)

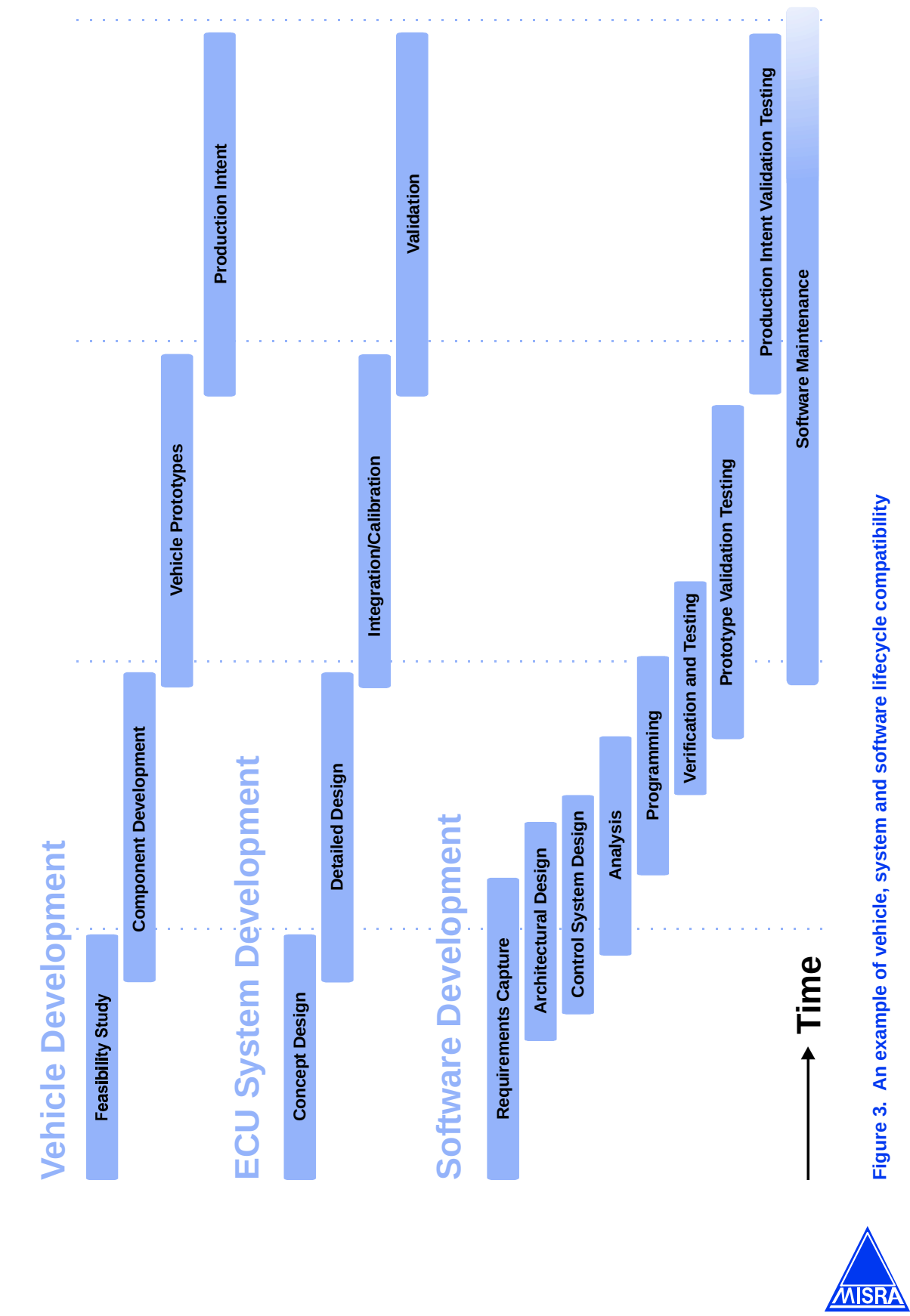


Figure 3. An example of vehicle, system and software lifecycle compatibility



Software lifecycle (continued)

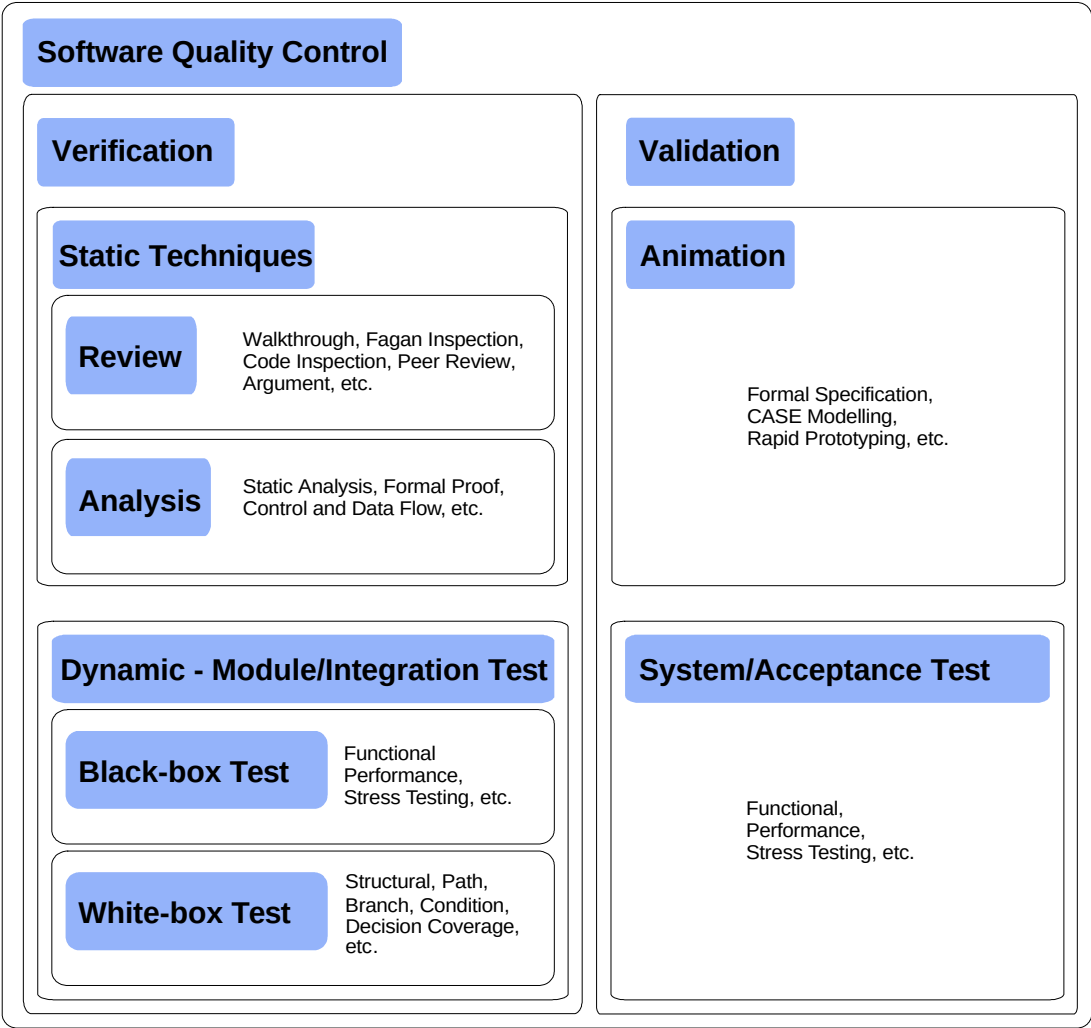


Figure 4. Software verification and validation



Software lifecycle (continued)

- 3.1.3.2 The purpose of software verification [6] is to ensure that each phase of the software development process maintains the attributes of the previous one without introducing errors. The outputs of each phase should be technically evaluated, by analysis and/or inspection with respect to its inputs, including standards and guidelines.
- 3.1.3.3 Safety verification is specifically concerned with the safety requirements and should proceed concurrently with the normal design and implementation process. Each phase of the development should also be verified with respect to safety before progressing on to the next development phase.
- 3.1.3.4 A verification plan should be established covering each phase of the software lifecycle.
- 3.1.3.5 The purpose of software validation [6] is to determine whether the software satisfies its intention and mainly consists of performing tests, but it should also involve analysis and mathematical techniques where appropriate. Full validation of the software cannot usually take place until it has been integrated with the rest of the system and often the vehicle.
- 3.1.3.6 Safety validation should be carried out to confirm specifically that all the safety requirements have been met.
- 3.1.3.7 A validation plan should be established for the software, system and vehicle.
- 3.1.3.8 To be effective, it is recommended that verification and validation are carried out by a person or team exhibiting a degree of independence from the design and implementation appropriate to the integrity requirements of the software and system.

3.1.4 Assessment

- 3.1.4.1 It is recommended that a company nominates a suitably qualified [19] and independent assessor and/or auditor to perform product assessment, in order to act as an advocate for the level of confidence in the safety delivered to the end customer [2].
- 3.1.4.2 An assessment process should be performed to demonstrate that the risks associated with the final system are at an acceptable level [14].
- 3.1.4.3 The higher levels of integrity should require greater independence of assessment in order to ensure that there is no bias from the development team, nor misplaced pressure from management.



Software lifecycle (continued)

3.1.4.4 The degree of independence [14] should be achieved by one or more of:

- different person
- different department or section
- different company or division.

3.1.4.5 The degree of confidence in a piece of software depends on the quality and the extent of the available information. This should cover the specification, design, implementation and testing of the software.

3.1.4.6 Preparing for assessment is a task that should begin at the start of a project; this also avoids the possibility of arriving at the point of assessment and finding that a vital piece of information has been omitted or overlooked. Early liaison with the assessor is recommended.

3.1.5 Reuse

3.1.5.1 “Off the shelf” or commercially available components containing software to be used in a project should be assessed to the required integrity level, and such reuse of existing software should be considered on a case by case basis.

3.1.5.2 The software that is intended for reuse should be fully compliant with the requirements of the new system that it is intended to be used in.

3.1.5.3 Software that is intended for reuse, as with all software, should have a standard of documentation such that its functions and interactions are easily understood by people who may not have been the original authors.

3.1.5.4 Software that has performed faultlessly for a number of years in the field may well offer a greater level of confidence in the final product than a wholly new program, provided it is used within the limitations of its design and is not modified.

3.2 Integrity

3.2.1 Introduction

3.2.1.1 Each system in a vehicle needs to have a level of integrity because there are requirements that it should not cause:

- harm to humans
- legislation to be broken
- undue traffic disruption
- damage to property or the environment (e.g. emissions).
- undue financial loss to either the manufacturer or the owner.



Software lifecycle (continued)

3.2.1.2 By achieving the required level of integrity, the risks associated with each of the requirements listed above should be reduced to an acceptable level [14]. Higher levels of integrity are achieved by the increased use of specialized methods and design requirements. This leads to an increasing cost, both of the development and of the end product.

3.2.1.3 The use of integrity levels permits a reasoned argument for assessing the degree of confidence that one should have in a system, which is commensurate with the inherent risk associated with the system function.

3.2.1.4 The recommendations and processes described in this section seek to enable a developer to:

- (a) analyse a system to determine its integrity level.
- (b) adopt a suitable development process in order to achieve the confidence level required in the software.

This ensures that the measures taken are both necessary and sufficient for the system being designed.

3.2.1.5 The system design may reduce the integrity level required for particular subsystems. For example, a failure mode set to a safe state by a hardware interlock of a suitable integrity level would remove that potential failure from the software hazard list. The hazard analysis therefore needs to be maintained throughout the development.

3.2.1.6 Once an integrity level for the software has been determined, an appropriate development approach should be defined, in order to gain the required confidence in the software.

3.2.1.7 The higher levels of integrity require more information and more rigorous application of software engineering techniques.

3.2.2 Safety analysis

3.2.2.1 A preliminary safety analysis [20] should be carried out as part of the requirements analysis phase, which becomes more detailed as the design progresses. Safety analysis is highly iterative and should be considered as a continuous process during requirements analysis and design [2].

3.2.2.2 A preliminary safety analysis should be performed early in the lifecycle to determine:

- the hazards associated with potential failures of the system
- the high level safety requirements necessary to reduce the risk associated with each hazard to an acceptable level [14]
- the initial integrity level.



Software lifecycle (continued)

- 3.2.2.3 The reasons for discounting potential hazards that are the result of extremely unlikely scenarios should be recorded during the preliminary safety analysis.
- 3.2.2.4 Once identified, each hazard should be analysed to determine its controllability category. The controllability approach was developed by the project DRIVE Safely [21].
- 3.2.2.5 The hazard with the highest controllability category will determine the integrity level of the system. This hazard analysis should be followed down the hierarchy to each system component (including software) to determine the integrity level required for each component.
- 3.2.2.6 The safety analysis is based on a hazard analysis using categories of controllability that are defined as follows:

Uncontrollable: This relates to failures whose effects are not controllable by the vehicle occupants, and which are most likely to lead to extremely severe outcomes. The outcome cannot be influenced by a human response.

Difficult to control: This relates to failures whose effects are not normally controllable by the vehicle occupants but could, under favourable circumstances, be influenced by a mature human response. They are likely to lead to very severe outcomes.

Debilitating: This relates to failures whose effects are usually controllable by a sensible human response and, whilst there is a reduction in safety margin, can usually be expected to lead to outcomes which are at worst severe.

Distracting: This relates to failures which produce operational limitations, but a normal human response will limit the outcome to no worse than minor.

Nuisance only: This relates to failures where safety is not normally considered to be affected, and where customer satisfaction is the main consideration.

- 3.2.2.7 Controllability applies to the ability of the vehicle occupants, not necessarily the driver, to control the safety of the situation following a failure. Systems for which the “occupants” may not necessarily be the driver include electric windows and electric seats.
- 3.2.2.8 The main steps involved in the determination of an initial integrity level are as follows (see Figure 5):



Software lifecycle (continued)

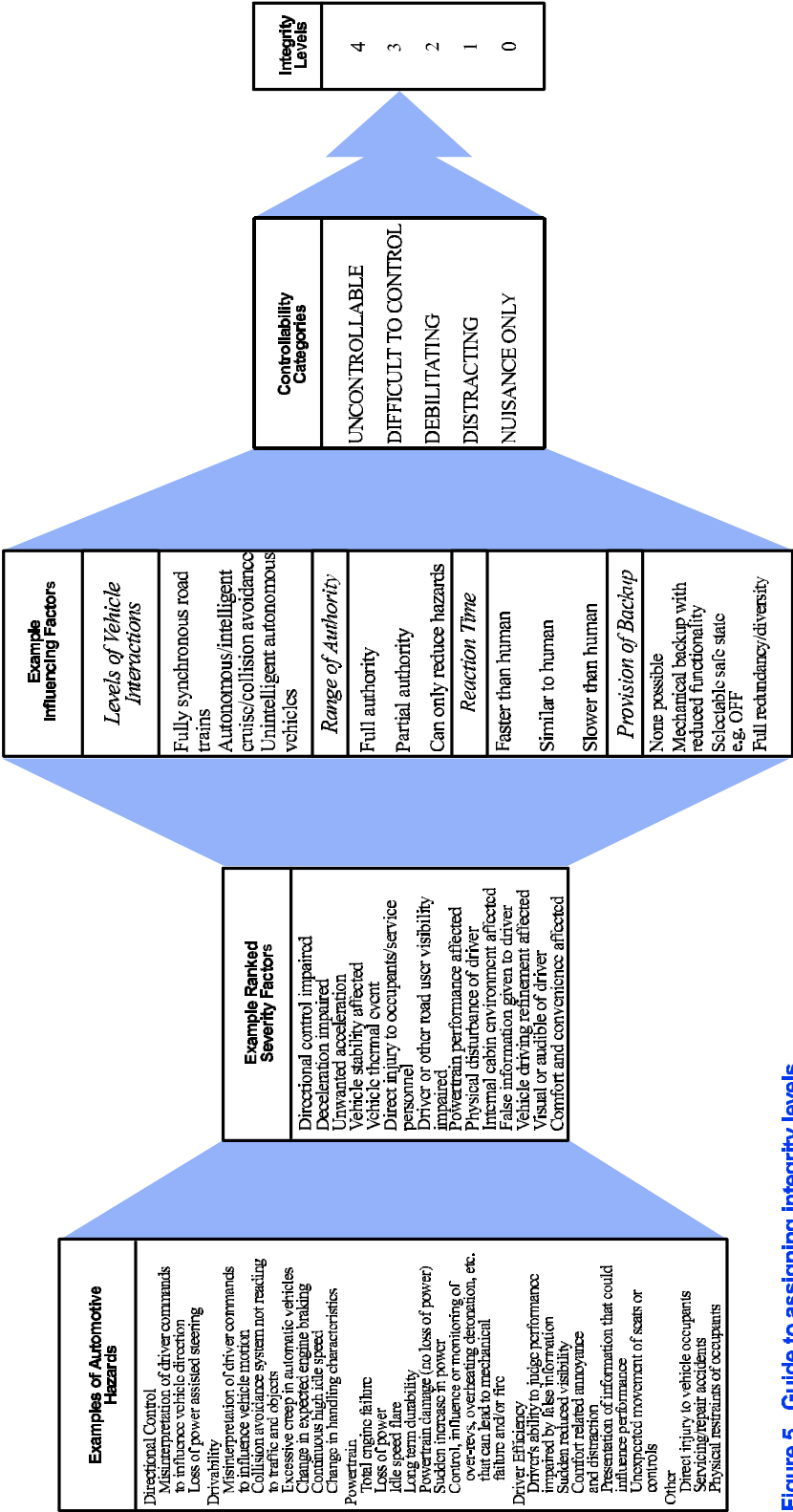


Figure 5. Guide to assigning integrity levels

