Werner Dirlewanger

Measurement and Rating of Computer Systems Performance and of Software Efficiency

An Introduction to the ISO/IEC 14756 Method and a Guide to its Application

kassel university press

Bibliographic information published by Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at http://dnb.ddb.de

ISBN-10: 3-89958-233-0
ISBN-13: 978-3-89958-233-8

Cover layout: Bettina Brand Grafikdesign, München
Printed by: Unidruckerei, University of Kassel
Printed in Germany

Preface

The performance of a data processing system is one of its most significant properties. For the user of such a system, a critical question is whether its performance will be adequate for the intended application. It is therefore desirable to be able to measure the performance of a data processing system. This question arises for all types and sizes of data processing systems. However, the measurement is difficult because of differing types, sizes and the high complexity of systems. A great variety of methods have been proposed and are being used to describe and measure performance. Each method was developed for a specific data processing system type and its use in a specific environment. Each has advantages and disadvantages. An additional problem is that the results of the different methods are not comparable.

To solve these problems ISO has developed a new method which is applicable for a wide range of data processing system types and applications. It is presented in the International Standard ISO/IEC 14756 "Information Technology - Measurement and rating of performance of computer-based software systems". Although this standard describes the method, it is not a tutorial. A textbook is desirable both for performance experts and beginners, and academic users. The aim of this book is to supply this need. On the one hand it introduces the latest techniques of computer performance measurement and of measuring software (run time) efficiency contained in the standard. And on the other hand it is a guide on how to apply the standard. However, you are recommended to buy the original ISO standard and read it in parallel.

This book focuses on measurement. It is not intended to be a general or broad overview of the wide field of all aspects of performance evaluation. For instance modelling, queuing techniques and simulation are not explained (many good books are available for those fields, for instance [ALLEN01], [BOLCH01], [JAIN01]). But the book discusses its own field in depth: i.e. measurement of system performance and of software (run time) efficiency. Additionally the author gives his experiences of many series of ISO-type measurements, of data processing systems of all sizes, which he planned and performed as a consultant or which have been performed in his laboratories.

This book is structured as follows: in each chapter the principles and methods are first explained, then illustrated with examples. Additionally there are exercises using the simple ISO-type measurement system DEMO, a simple demonstration software. These exercises can be performed on any UNIX operating system. The book shows how to do this using the well known and inexpensive LINUX. All needed software is included on a CD supplied with the book and is published by the author under GNU license. The demonstration software DEMO is not intended to be a tool for performing professional measurements. It only enables the reader to realise all steps of the measurement procedure and to observe all details as they occur. DEMO is deliberately not highly automated. It works interactively. In its native mode each step of the measurement procedure has to be manually controlled in order to show the trainee what happens.

The reader can use this book in different ways. For instance, if he is only interested in seeing some of the basic ideas of system performance measurement of the ISO method, then he should only read the corresponding chapters. But should he, for instance, be interested in a deeper understanding of the method and its applications then it would be mandatory to read all chapters thoroughly and to perform all exercises at the end of the

chapters. Undoubtedly this will cause him to invest several hours of work, with possibly increased resources. It is up to the reader to decide what to do.

There is an additional benefit in using the DEMO demonstration software. As it shows in detail the principles of an ISO-type performance measurement tool, it helps industry and science in implementing an ISO-type performance measurement system for professional use. It must be stressed, however, that DEMO is only a demonstration tool on how an ISO-type measurement system works. It is neither a reference implementation nor a tool for professional measurement.

Although this book introduces the reader to modern performance measurement and to ISO/IEC 14756, it is not intended to replace this standard. The principles and methods are shown and explained, but for all normative details the reader must refer the original text of the standard. Should the reader be interested in mastering the complete mathematical presentation, this book can prepare him for more intensive study of ISO/IEC 14756.

The ISO standard uses conventional mathematical terminology and presentation. The author was tempted to do so in his book. Although he prefers this style, he resisted the temptation in order to enable less mathematically trained readers to follow the material.

ISO contributed an essential support to the realisation of this book by granting permission to reproduce, in the Appendix of this book, the workload examples specified in Annex "F" of the standard. I am most grateful for this permission.

Many individuals have stimulated and supported my work in preparing this book. I am really indebted to all of them. My gratitude goes to Annette and Robin Calkin for their many strenuous hours of proof reading and for their many recommendations to the text and to my children Christine and Christian for designing the graphics. My thanks also go to Reinhold Schlüpmann for his encouragement and editorial support in the development of the ISO standard; to Sascha Gortzewitz and Piotr Szulc for checking and testing the ISO method in my laboratories, and implementing, checking and testing the ISO workloads and for their contributions to the initial development of the DEMO software; to Eberhard Rudolph for writing Section 14.8 on applying the ISO method to function point measurement; and to Wolfgang Eichelberger, Reinhold Grabau and Stefan Kucera for their contributions when developing predecessors of the ISO method, putting them into practice and building up experience before publishing the ISO standard.

To all these people I extend my sincerest thanks. But finally I would like to especially thank my wife and children for their support and understanding when I was writing this book. It is dedicated to them in recognition of their patience during my countless hours absorbed in preparing the manuscript.

Werner Dirlewanger

Contents

Preface 1

Contents 3

Chapter 1 General basics 11

1.1 Computer performance evaluation and benchmarking 11
1.2 System performance compared to component performance 12
1.3 ISO scope of system performance 12
1.4 Measurement of computer performance compared to prediction methods 14
1.5 What is rating of performance and why is it needed ? 14
1.6 Basic principles and philosophy of ISO/IEC 14756 15
1.7 Overview of ISO/IEC 14756 16
1.8 Exercises 17

Chapter 2 The ISO workload 19

2.1 The view of the ISO workload 19
2.2 Basic ideas of the ISO workload description method 21
2.3 Explanation of the terms "activity, "activity type", "task" and "task type" 22
2.4 Explanation of the terms "chain" and "chain type" 23
2.5 Explanation of the timeliness function 24
2.6 The basic parameters of an ISO-type workload 25
2.7 The user behaviour parameters 26
2.7.1 The activity type values 27
2.7.2 The task type values 28
2.7.3 List of timeliness function values 28
2.7.4 List of chain type definitions 29
2.7.5 The relative chain frequencies 29
2.7.6 Preparation time mean values 31
2.7.7 Preparation time standard deviation values 32
2.8 Application programs, their data and computational results 33
2.9 The advanced parameters of an ISO-type workload 33
2.9.1 Computational results 33
2.9.2 Precise working of the RTE 33
2.9.3 Statistical significance 34
2.10 Short summary of the contents of an ISO workload 35
2.11 Exercises 35

Chapter 3 The measurement experiment 37

3.1 Principles of operation of the ISO-type user emulator (RTE) 37
3.2 Dynamic task generation versus pregenerated task lists 40
3.3 The three phases of a measurement run 42
3.4 The logfile (measurement result file) 43
3.5 Storing the computational results 44

3.6 Some random generation methods 44
3.6.1 Generation of random chain type numbers 44
3.6.2 Generation of random preparation times 45
3.6.3 A practical problem with finite random sequences 46
3.7 Exercises 46

Chapter 4 Validation of the measurement results 51

4.1 Validation of the computational results of the SUT 51
4.2 Validation of the correctness of the working of the RTE 52
4.2.1 Three criteria 52
4.2.2 The first criterion: Checking the relative chain frequencies 52
4.2.3 The second criterion: Checking the preparation mean times 53
4.2.4 The third criterion: Checking the standard deviations of the preparation times 54
4.2.5 Remarks 55
4.3 Checking the statistical significance of the measurement results 55
4.3.1 Rationale for this check 55
4.3.2 The test 56
4.3.3 Application of the sequential test 57
4.3.4 Fast computation of mean value and variance 58
4.4 Summary of the validation procedure 59
4.5 Exercises 61

Chapter 5 Computing the ISO performance values from the measurement result file (logfile) 63

5.1 Overview of the ISO performance terms 63
5.2 The "total throughput vector" B 63
5.3 The "mean execution time vector" T_{ME} 64
5.4 The "timely throughput vector" E 64
5.4.1 The principle of E 64
5.4.2 Computing e(j) 65
5.4.3 The "timely throughput vector" E 67
5.5 Exercises 67

Chapter 6 The Urn Method 69

6.1 General 69
6.2 Introduction to concept of individual rating intervals 69
6.2.1 Defining the individual rating intervals 69
6.2.2 Modifying the computation of performance values 70
6.2.2.1 Computation of B (total throughput) 70
6.2.2.2 Computation of T_{ME} (mean execution time) 71
6.2.2.3 Computation of E (timely throughput) 71
6.2.3 Modifying the definition of the end of the SR 72
6.2.4 Overlapping of the individual RIs 72

6.3 Explaining the concept of "urns" 72
6.3.1 Toleration of the Urn Method by ISO/IEC 14756 72
6.3.2 The urns 73
6.3.2.1 Generating chain sequences 73
6.3.2.2 Generating preparation time sequences 73
6.3.3 Generation of a set of preparation time values for filling a preparation time urn 74
6.4 Experiences from applying the Urn Method 75
6.5 Formal and detailed description of the modifications for computing the performance values 76
6.5.1 Total throughput vector B 76
6.5.2 Mean execution time vector T_{ME} 76
6.5.3 Timely throughput vector E 77
6.5.4 Explanations 77
6.6 Exercises 78

Chapter 7 Rating the measured performance values 79

7.1 The principle of the ISO rating 79
7.2 The ISO theoretical reference machine 79
7.3 Computation of the reference performance values 80
7.3.1 Computation of T_{Ref} 80
7.3.2 Computation of B_{Ref} 81
7.4 Throughput rating 83
7.5 Rating the mean execution times 83
7.6 Timeliness rating 84
7.7 Overall rating 85
7.7.1 General ISO rule of rating 85
7.7.2 Extended ISO rating rule 85
7.8 Exercises 86

Chapter 8 The performance measure N_{max} 87

8.1 Maximum number of timely served users (N_{max}) 87
8.2 Incrementing the number of users 87
8.3 Measurement series 88
8.4 Acceptable tolerances of N_{max} 89
8.5 Experiences from various measurement series 89
8.6 Exercises 92

Chapter 9 Summary of the ISO system performance measurement method 95

9.1 The steps of an ISO-type measurement run 95
9.1.1 Step 1: Specification of the workload 96
9.1.2 Step 2: Installation of the applications on the SUT 96
9.1.3 Step 3: Connecting the SUT to the RTE 96
9.1.4 Step 4: Loading the RTE with the WPS 96
9.1.5 Step 5: The measurement run 96
9.1.5.1 Basic form of measurement with common rating intervals 96
9.1.5.2 Measurement with individual rating intervals 97
9.1.6 Step 6: Checking the correct working of the SUT 98
9.1.7 Step 7: Checking the correct working of the RTE 98
9.1.8 Step 8: Checking the statistical significance and the RI overlap 98
9.1.9 Step 9: Calculating the performance values 98
9.1.10 Step 10: Calculating the rating values 100

9.2 Computing the measurement results 100
9.2.1 Calculation of the performance values 100
9.2.2 Calculation of the rating values 101
9.3 The measurement report 102
9.3.1 Principles 102
9.3.1.1 Completeness 102
9.3.1.2 Detailed report 102
9.1.3.3 Clarity 102
9.1.3.4 Data formats 102
9.3.1.5 The storage medium 103
9.3.1.6 Reproducibility 103
9.3.2 Suggested contents of the measurement report 103
9.4 Recommendation for the documentation of a measurement run 105
9.4.1 Measurement operator's protocol 105
9.4.2 Archiving the measurement files 106
9.4.3 Safekeeping period 106
9.5 Reproducibility of measurement results 107
9.6 Exercises 107

Chapter 10 Measurement of software (run time) efficiency 109

10.1 A hierarchical model for information processing systems 109
10.2 The reference environment and the term run time efficiency 110
10.3 The measurement procedure and measures of software efficiency 110
10.3.1 The measurement procedure 110
10.3.2 Run time efficiency terms related to task types 111
10.3.3 A software run time efficiency term related to the performance measure N_{max} 113
10.3.4 Comparison of the two methods 113

10.4 Examples 114
10.4.1 Example 1: Application software efficiency 114
10.4.1.1 The measurement environment 114
10.4.1.2 The task-oriented software efficiency values 114
10.4.1.3 The N_{max} oriented software efficiency value 116
10.4.2 Example 2: System software efficiency 116
10.4.2.1 The measurement environment 117
10.4.2.2 The task-oriented software efficiency values 117
10.4.2.3 The N_{max} oriented software efficiency value 119
10.5 Exercises 120

Chapter 11 The ISO workloads 123

11.1 Purpose of the workloads and format 123
11.2 The Simple Workloads 127
11.2.1 General 127
11.2.2 SIMPLOAD1 127
11.2.3 SIMPLOAD2 127
11.2.4 SIMPLOAD3 128
11.3 The Computer Centre Workloads 128
11.3.1 General 128
11.3.2 COMPCENTER1 128
11.3.3 COMPCENTER2 128
11.3.4 COMPCENTER3 129
11.4 Migration of ISO workloads to other operating systems 129
11.5 Important details for ISO workload migration 130
11.5.1 Workload COMPCENTER1 130
11.5.1.1 Introduction 130
11.5.1.2 The logical steps of the OSCPs of COMPCENTER1 131
11.5.1.3 Some explanations 132
11.5.2 Workload COMPCENTER2 133
11.5.2.1 Introduction 133
11.5.2.2 The logical steps of the OSCPs of COMPCENTER2 134
11.5.2.3 Some explanations 134
11.5.3 Workload COMPCENTER3 135
11.5.3.1 Introduction 135
11.5.3.2 The logical steps of the OSCPs of COMPCENTER3 135
11.5.3.3 Installation of the workload COMPCENTER3 on the SUT 135
11.5.4 Migration examples of ISO workloads 135
11.6 Exercises 136

Chapter 12 Creating an individual ISO-type workload 137

12.1 Activity types and their representatives 137
12.2 Timeliness functions 137
12.3 Task types 138
12.4 Chain types 138
12.5 Chain probabilities and user types 139
12.6 Preparation times ("think-times") 140

12.7 The WPS 140
12.7.1 Recording the values of the WPS in a text file 140
12.7.2 Recursive improvement of the WPS 141
12.8 The application software 142
12.9 The advanced parameters 142
12.10 Preparation 142
12.11 Migration of an individual workload to a different operating system 142
12.12 Exercises 143

Chapter 13 Organisation and management of an ISO-type measurement project ..145

13.1 Deciding on the goals of the measurement project 145
13.1.1 List of performance measurements goals 145
13.1.2 List of software run-time efficiency goals 146
13.2 Defining the responsibilities 147
13.3 Assessing the costs of the project 147
13.4 The project schedule 148
13.5 Making the workload available 148
13.6 Making the SUT operational and tuning it 148
13.7 Choosing an ISO-type RTE having sufficient performance 149
13.8 Performing the measurement 149
13.9 Computation of the performance values and rating values 150
13.10 Audit 150

Chapter 14 Miscellaneous aspects 151

14.1 Measurement using a real workload 151
14.2 Measurement using automated sample users 151
14.3 Measuring single-user systems 152
14.4 Hidden batch jobs in online transaction processing 153
14.5 A distributed RTE 154
14.6 Life cycle of ISO-type workloads 154
14.6.1 Workload definition and documentation 154
14.6.2 The RTE 154
14.6.3 Type, architecture, manufacturer of the SUT 155
14.6.4 Power of the SUT 155
14.6.5 Applications contained in the workload 155
14.6.6 Final remarks 155
14.7 Reliability aspects 156
14.8 Conversion of non ISO-type workloads to the ISO data model 156
14.8.1 Candidates for conversion 156
14.8.2 The conversion procedure 156
14.8.3 Examples of conversions and sketches of some individual workloads 157
14.8.3.1 The classic non-multiprogramming batch benchmark 157
14.8.3.2 The classic multiprogramming batch benchmark 160
14.8.3.3 The "Debit-Credit-Test" for OLTP systems 161
14.8.3.4 The SPECweb99 test for internet servers 166
14.8.3.5 The KS-Web workload for intranet servers 167
14.8.3.6 The Kassel data warehouse workload 167
14.8.3.7 An ERP workload 167

14.9 Example structure of an ISO-type measurement system 169
14.9.1 The example structure 169
14.9.2 Short descriptions of the modules 171
14.9.3 Some comments on the actual implementation of DEMO 173
14.10 Applicability of the ISO method for measuring component performance 175
14.11 Short comparison of some other methods with the ISO method 176
14.11.1 Incomplete list of commonly used system performance measurement systems 176
14.11.2 Short comparison 178

14.12 Applying ISO/IEC 14756 to Function Point Measurement 179
(written by Eberhard Rudolph)
14.12.1 Overview 179
14.12.2 Activated Function Points (AFP) 179
14.12.2.1 Deriving AFP 180
14.12.2.2 AFP Example 180
14.12.3 Using AFP with ISO/IEC 14756 180
14.12.3.1 SAP-R/2 measurement 181
14.12.4 Limitations 183
14.12.5 Opportunities 183

Appendix A: CD as a part of the book 185

- ISO/IEC original workloads
- Supplement to ISO/IEC 14756 (logical steps of OSCPs)
- Two ISO workloads converted to LINUX
- Sketch of an ISO-workload converted to NT4.0
- Sketches of ISO-type individual workloads (ERP, ExternalDWH, KS-Web, Mainframes, Web99)
- The measurement System DEMO (software and manual)
- Detailed documentation of a measurement using DEMO
- Solutions of exercises
- Files of the exercises

References 187

Abbreviations 189

Symbols 191

Index 199

1 General basics

This book introduces the modern principles of computer performance measurement and of measuring software run time efficiency. It refers especially to the method defined in the ISO/IEC 14756, see [ISO14756], and it is a guide of how to apply this standard. Finally it helps programmers when designing and implementing software tools for measurements.

Note: The author makes no warranty of any kind with regard to the material of this book and shall not be liable for defects contained herein or for incidental or consequential damages with the use of this material, either expressed or implied.

1.1 Computer performance evaluation and Benchmarking

What is "performance" ? We used to say: our computer has a high performance. In more detail one could say: it is fast and reliable; it forgets anything; it makes no errors; it is always available for work; it can do many different tasks; it is never tired. Looking further at these statements we find that the characteristics of a computer system can be divided into three classes.

First class: This class describes what the computer is able to do. It is the set of activities, the correctness of operation and of the computed results and, additionally, user friendliness (ergonomics).

Second class: This class describes how stable and consistent the computer is in its operation. It refers to the reliability of operation in the widest sense.

Third class: This class refers to the speed of operation. One aspect is the speed of executing the tasks, i.e. the time for delivery of the task results. Another aspect is the number of tasks which can be executed in a given time.

The classification assumes that a computer system is not restricted to a single-user stand-alone machine, such as a historical personal computer. Much more it is a general information processing system (IP system) including multi-user, multi-processor, or networking architectures.

The third class cited above means computer performance in the proper sense, i. e. the speed of operation. This book focuses on this meaning of performance.

What is performance evaluation ? This activity involves all evaluation methods of performance of data processing systems where performance is expressed by numbers and not only by words like high or low. Additionally evaluation can include the assessment of how good the performance satisfies the user requirements.

This book focuses on numerical performance values, determined by measurement and their assessment. Assessment also results in numerical values.

What is benchmarking ? The word benchmark is taken from the marks showing measures like feet or inches on the work bench of former craftsmen. In the data processing field

several principles are in use for performing an evaluation. Some of them use a model of the IP system. Another principle is to investigate the real IP system by loading it with concrete applications, programs and data. Every method of determination performance values in such an experiment with the real system under test is benchmarking. This book is concerned with benchmarking and focuses on the ISO method.

1.2 System performance compared to component performance

The obvious way for characterising the performance of an IP system is simple. First decide which is the most significant component of the system, for instance the central processing unit (CPU). Then define a performance term, for instance the number of executed instructions per time unit, or a set of performance terms. Determine the values of this performance terms and take it (or them) as the performance of the IP system. This is the principle of "component performance".

This way may have been appropriate in earlier times when the IP systems had been simple single-user type machines having for instance, no multiprogramming features, no networks and only one central processor. In contrast with these early computers a present day IP system is much more complex. It is no longer sufficient to describe its performance by values of its most important hardware component. The IP system has to be seen in its entirety. Suitable performance terms have to be defined with regard to it. This consists of all system and network components producing the term "system performance". An important practical experience is that it is usually not possible to compute the system performance values from the component performance values. Therefore a separate method for determinating the system performance is needed. This book focuses on system performance and not on component performance.

Note: The component performance values and the system performance values often are named "external performance values". Contrary to these terms are the "internal values" as, for instance, the CPU utilisation, the storage utilisation, the utilisation of data busses, the length of job queues and the multiprogramming factor. In the past, those internal values were often used for performance values. They are no longer suitable. They only describe internal load situations of IP system components. In contrast to the external values they do not characterise the speed of operation. Internal values are not used in the ISO method and are not considered in this book. ■

1.3 ISO scope of system performance

The method defined in ISO/IEC 14756 [ISO14756] follows strictly the idea of system performance as described above. The system is the set of all co-operating hardware, software and network components. This set is regarded as a black box, which is connected via the set of its interfaces to its users (see Fig. 1-1). The users are typically humans, but some of them can be machines which submit tasks to the system via an interface.

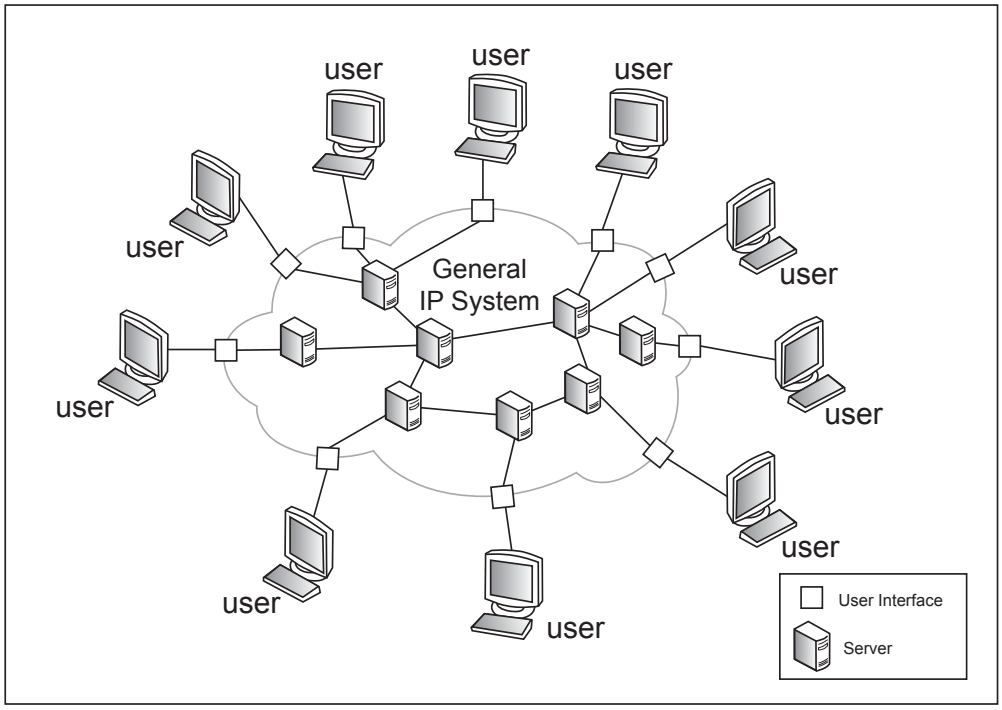


Fig. 1-1 The general IP system and its user entirety

1.4 Measurement of computer performance compared to prediction methods

The field of computer performance evaluation has the three subfields measurement, simulation and modelling.

Measurement means carrying out a real experiment with the real IP system operating in real time with real users. A monitoring feature records all necessary data during the experiment. Performance values are computed from the recorded data.

For simulation a mostly simplified functional model of the IP system and its users is developed. A computer program is then written which runs the model. This program may run in slow motion, in real time or time-lapse mode. It does not matter which one of these three modes is used. All necessary data during this simulated run are recorded by a software monitor. Performance values are computed from the recorded data.

For modelling a very simplified functional model of the IP system and its users is developed. From this a mathematical model is derived by means of queuing theory. This model can be analysed by solving the so-called state equations merely numerically. But sometimes also the explicit formulae of the interesting performance terms are found. Then the performance values can be computed by use of the formulae.

In contrast to this in a measurement the real IP system is investigated and tested. Simulation and modelling use only models of the system under test. Therefore the last two methods deliver performance values of models. These are estimated and not measured

values. Consequently simulation and modelling deliver only predictions of performance values. This book is not concerned with prediction methods. It focuses on real measurement (as represented by the ISO method).

1.5 What is rating of performance and why is it needed ?

The results of a performance determination, independent of whether it is done by a measurement method or by a prediction method, are performance values. They are values of physical properties of the IP system under test (SUT), i.e. physical values. They are not information about a more far-reaching aspect of using data processing systems. This is the question if a regarded IP system fulfils the performance requirements of its user entirety.

The requirements of the user entirety are primarily non numeric values such as "poor", "sufficient" or "excessive". We have to define which ranges of performance values correspond to each of these three non numeric values. I.e. we have to relate the non numeric values to the scale of the performance values. Entering the performance values of the SUT into this scale delivers the rating result, for instance "the performance is sufficient". This is shown in Fig. 1-2 .

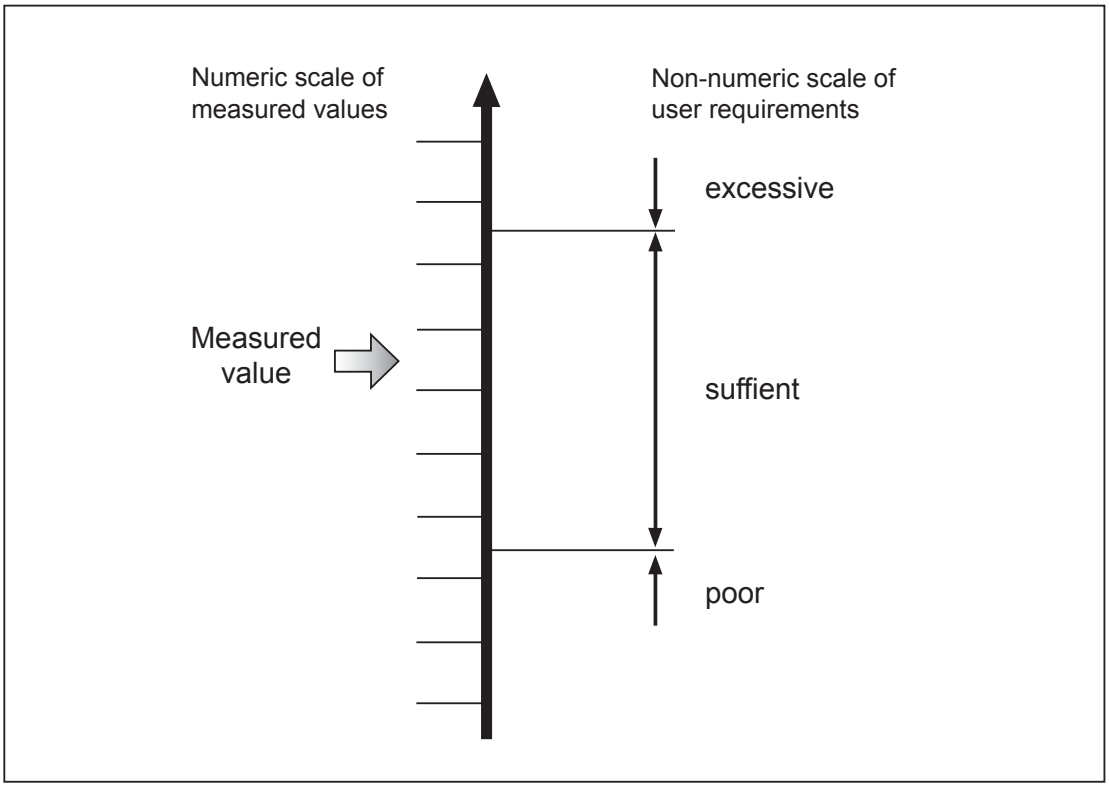


Fig. 1-2 Rating of performance values

The ISO method (which is the focus of this book) contains a comfortable rating procedure. It is explained further in chapter 7.

1.6 Basic principles and philosophy of ISO/IEC 14756

It is not the goal of this section to describe in detail all the ideas of ISO/IEC 14756. Only a selection of the more interesting principles is listed below.

1. ISO/IEC 14756 deals with system performance evaluation as defined in Section 1.2 and not with component performance. (As it will be shown in Section 14.10 a slightly modified method can be applied to component performance evaluation.)

2. ISO/IEC 14756 defines a measurement method but not a prediction method. (For "measurement" and "prediction" see Section 1.4).

3. ISO/IEC 14756 is more than a "benchmark". It is a set of co-ordinated and co-operating methods. The standard firstly defines the performance term for system performance; secondly it describes a method for defining workloads (workload data model); thirdly it defines a method for measuring the data processing performance of a defined IP system using a workload which is constructed according to the ISO workload definition; fourthly it defines a method for rating the measured performance. (For "rating" compare Section 1.5).

4. There is exactly one universal data model for defining workloads. It is the same for all possible workloads.

5. The ISO method does not represent a monolithic benchmark. Contrary to monolithic benchmarks the measurement method is isolated from the workload. Its terms and performance measures are the same for all workloads and tests.

6. Applying a defined workload, described according to the ISO method, and carrying out the measurement and the rating procedure, according to ISO, realises a benchmark. That means that an unlimited number of ISO benchmarks can exist as anyone can define an individual ISO-type workload.

7. The SUT is seen as a black box. Therefore types, operating systems, architecture etc. of the SUTs can be different. Nevertheless the measured performance values are objective and can be compared without difficulties. (Note: This could imply that the performance of a tiny old fashioned workstation or a personal computer can be compared to that of a huge data processing system or a computer network. This comparison is objective with respect to the used workload.)

8. The users in an ISO measurement are simulated by a simulator (remote terminal emulator, RTE) which is not part of the SUT. It is external. The principles of operation of this RTE are described and defined in detail in the ISO standard. This produces measurements that can be reproduced repeatedly and independent of the personnel carrying out the measurement.

9. The workload is input to the RTE which is table driven. Therefore one implementation of the RTE is in principle sufficient for all workloads. In practice typically one ISO type RTE is sufficient for a wide class of workloads. Only a few modifications are needed for other classes.

10. The ISO workloads are of the "ready for run" type. They realise no "high level benchmarks". This implies, compared to high level benchmarks (the application programs of which are detailed specified but not programmed), less work for preparing a measurement. In the case of the ISO-type workload being already adapted to the SUT operating system type the amount of work is in fact less.

11. There are strong verification rules for checking the technical correctness of a measurement run. Check criteria are specified for the correctness of the work of the RTE, for the correctness of the work of the SUT and for the statistical significance of the measurement results. Note: ISO/IEC 14756 is strictly technical and may not include legal aspects. Therefore no auditing is included and no pricing is included, which means that the ISO/IEC 14756 defines no rules for non-technical aspects, such as cost/performance values.

12. The response time requirements of the user entirety are a mandatory part of the workload. A good way of defining these requirements is given in the standard. It is the basis for realising the rating procedure of the performance values. It enables the computation of the performance values required by the user entity and the rating of the SUT as "poor", "sufficient" etc.

13. As the ISO standard is a principle for measurement of system performance it is also possible to measure the run time efficiency of software (see Chapter 10). This aspect is also described in the standard.

1.7 Overview of ISO/IEC 14756

The title of the standard "Measurement and rating of performance of computer-based software systems" is a little misleading. The author of this book feels that a more suitable title would be "Measurement and rating of performance of information processing systems and of software run time efficiency".

ISO/IEC (see [ISO14756]) consists of 50 pages text and a CD. The text consists of a short foreword and introduction, three sections, four normative annexes and two informative annexes. The CD contains the annexes "D" (programs) and "E" (workload examples) in machine readable format. However the 50 pages of text are also on the CD, in "pdf format", together with some small files containing copyright information and a file on how to unpack compressed files.

The standard is not available free of charge. For reproduction or utilisation apply to the publisher, ISO/IEC Copyright Office, Case Postale 56, CH 1211, Genève 20, Switzerland. Permission has been granted by ISO to quote or reproduce parts of the standard in this book.

Section 1 of the standard contains five paragraphs describing the scope of the standard, conformance, normative references, definitions and abbreviations and symbols.

Section 2 and Section 3 give the impression of having very different contents. But they are to some extent similar. Section 2 explains in four paragraphs the principles of measurement and rating and is to some extent an introduction to the methods used. The 6 paragraphs of Section 3 treat these methods in detail.

There are four normative annexes. "Annex A" specifies the principles of the RTE in detail. It is the basis for programming the kernel of an ISO-type RTE. "Annex B" summarises those mathematical formulae which are for simplicity not printed in the main part. "Annex C" specifies the format of the workload description. The short "Annex D" lists all mandatory data set entries of the logfile which have to be recorded during a measurement.

Additionally are there two non normative annexes: "E" and "F". "Annex E" contains programs. These programs are examples of how to implement some formulae and algorithms of the ISO method, which are perhaps a little too sophisticated. It may not be obvious how to program them. The programs are intended to be a help for programmers when implementing an ISO-type measurement system, but they do not realise a complete measurement system.

Note: A complete ISO-type measurement system is the DEMO system. It is found on the CD which is part of this book. The structure of a complete ISO type measurement system is explained in Section 14.9 . ■

Annex "F" contains six ISO-type workloads. In the standard they are declared to be examples. Although these six workloads are not normative, at least three of them (the so-called computer centre workloads) have proved to be suitable for professional performance evaluation. These three workloads are nearly identical to those workloads which are parts of the National German Standard DIN 66273 (see [DIN01]).

Measuring software (run time) efficiency is only briefly described in subparagraph "8.2" of ISO/IEC 14756. In comparison with the extensive work of defining the system performance measurement, the principle of software efficiency measurement is almost simple (see Chapter 10 of this book).

1.8 Exercises

Exercise 1: Making available a demonstration test bed

The ISO method cannot be performed manually. It needs a computer-aided tool, which is the RTE, and a suitable SUT.

Part 1
You need two Pentium compatible computers, each running a LINUX operating system. One is used as the platform of the RTE, the other for the SUT. These two machines have to be connected by an Ethernet line with a speed of at least 10 kbits/second. The RTE machine should have a CPU with a speed of at least 500 Mhz, and be significantly faster than the SUT machine.

Part 2
Create in the RTE a user named "operator" and in the SUT a number users (for instance 25) named "user1", "user2",..... . Make sure that the "operator" can access via the network all users of the SUT with the UNIX commands "telnet", "ftp" and "rsh".

Exercise 2: Installing the RTE software DEMO

This book includes a complete ISO-type performance measurement system: the DEMO. This software is available under GNU License and is contained on the CD-ROM as part of this book. The DEMO is not a professional system. See Section 14.9.3 for its limitations. But it is suitable for learning and teaching. Install DEMO on the RTE machine according to the instructions on the CD-ROM.

Remark: Check if your computer uses a LINUX operating system release needed by DEMO (see file CD/DEMO-20/release.txt). If not, a few alterations will be necessary to DEMO.

Exercise 3: Installing the system software components on the SUT

Compilers (including those for C, FORTRAN77, COBOL ANS85) have to be installed and also the editor "ed". They are needed for running the ISO workload examples in the following chapters of this book. For economy you are recommended to use low cost software (e. g. free or GNU licensed), which is suitable for running the workloads.

Exercise 4: Documentation of the test bed configuration

Write down the documentation of the hardware and software of the test bed.

Solutions

For solutions see file

```
CD/Solutions/Solutions-Section1-8.pdf .
```

2 The ISO workload

2.1 The view of the ISO workload

The SUT, naturally, remains idle if there are no demands on any active data processing tasks. It comes into operation only if users become active. Then subsequently it "is workload on the system". But what is "workload" ? Looking to the literature shows many meanings of workload. They have a strong dependence of the view. There are three classes of views of workload.

- The internal system view
The workload is described by a set of terms, as for instance

- CPU utilisation
- average storage usage
- I/O rate
- length of queues

The point of view is located in the system (see Fig. 2-1). How do we compute these values from a defined user community ? This attempt fails in praxis.

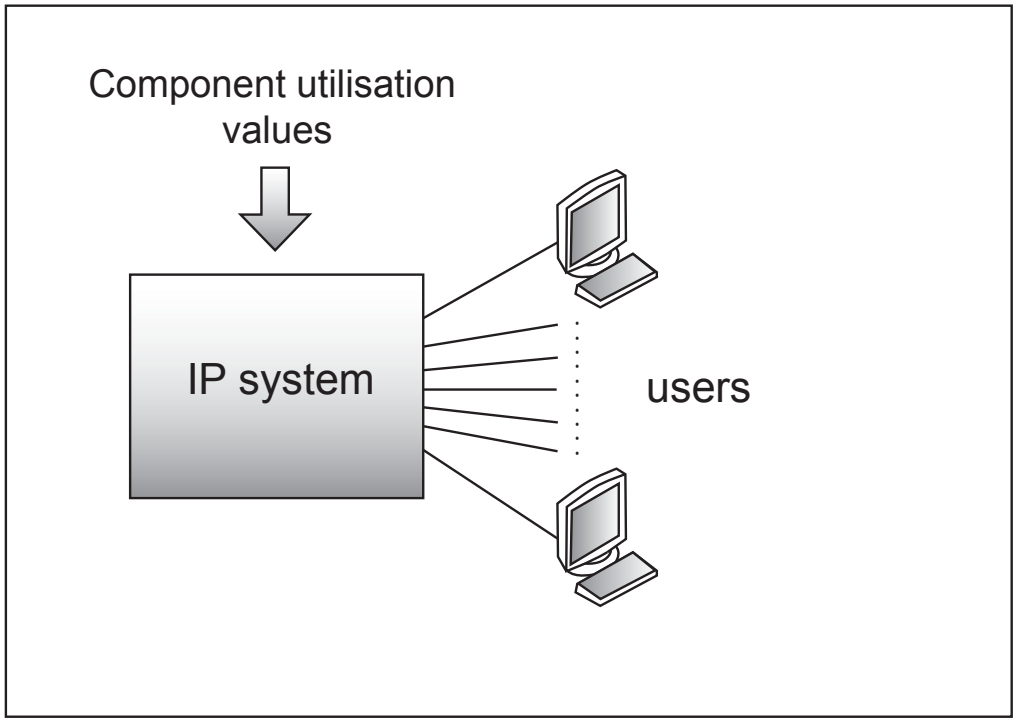


Fig. 2-1 The system internal view of workload

- The user interface view
The workload is described by terms like

- rate of interactive commands
- rate of submitted batch tasks
- rate of OLTP tasks

(OLTP = online transaction processing)
The point of view is located at the interface between users and the system
(see Fig. 2-2). How do we compute these values from a defined user community ?

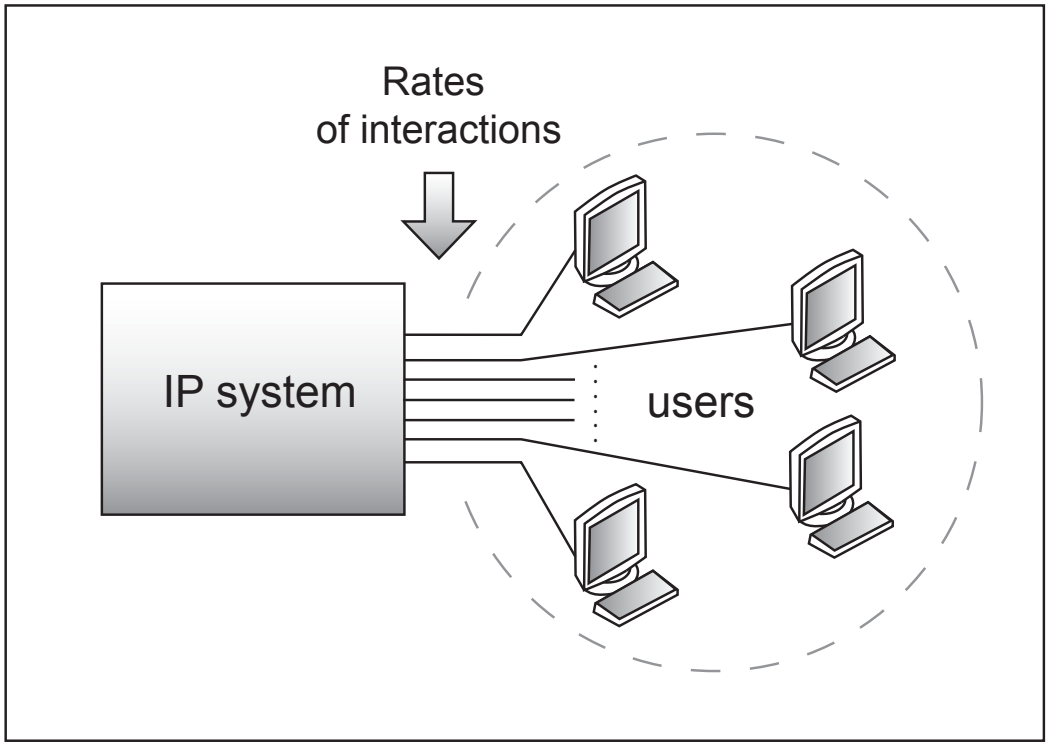


Fig. 2-2 The user interface view of workload

Among others there is the problem of "feedback" as follows. Faster users mean shorter think times. Slower users mean longer think times. Regarding a defined SUT the throughput increases or slows down when the users are faster or slower. The response times of the SUT increase or decrease when the users work faster or slower. All throughput values (as quoted above) depend on the users. The situation is analogous to economics. Sales increase if customers place their orders promptly; they decrease if customers delay placing their orders. The feedback is a severe problem of the interface oriented workload view. The computation of this type of workload from a defined user community could be successful by setting rough estimates (including many assumptions on the time behaviour of the SUT). But in general it seems to be unsolvable.

- The user-oriented view
In this view the workload is the set of all users and their application programs and data (see Fig 2-3). This workload definition is independent of the SUT properties. The SUT has no influence on the workload terms. The problem how to compute the values of the workload does not exist. The values of the workload terms are directly determined by the user community. The ISO standard uses a user-oriented view of the workload.

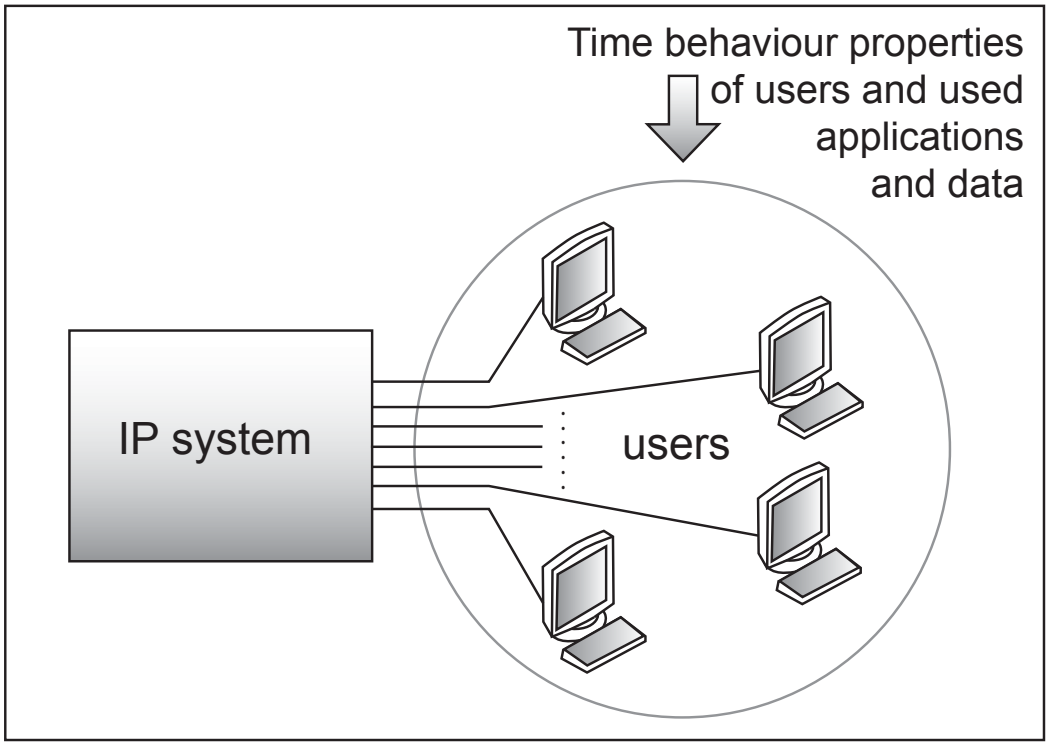


Fig. 2-3 The user-oriented view of workload

2.2 Basic ideas of the ISO workload description method

The ISO workload is defined as follows:

> Workload is the set of the time behaviour properties of the entirety of all users (of the data processing system under test) and all the used application programs and data.

A data model is used to describe the properties of the set of all users. This set is called user entirety. This model is the same for all workloads. It is explained here. (Note: In the ISO standard the specification of this model is distributed over several paragraphs. A summary is found in "Annex C" of ISO/IEC 14756.) In the data model the user entirety consists of a set of users. Each user submits tasks (to the SUT) independently from other users. Each submission is preceded by a random think time. The think times, in the standard called preparation times, are defined by a mean time and standard deviation. The values of mean time and standard deviation can depend on the user and the type of submitted tasks. The data model takes into account that the submission of a task of a defined type can be preceded and followed by defined (and not randomly chosen) task types. This allows the definition of fixed task sequences, called task chains. For each user the chains used by him have to be defined and listed. The length of a chain is either the number of contained tasks or may be one task, in which case there is no predecessor and no follower task. The users submit chains in a random order but with defined probabilities of each chain type. For each

user the relative frequencies of submitting chain types have to be defined. These relative frequencies may differ for each user.

2.3 Explanation of the terms "activity", "activity type", "task" and "task type"

The basic element of the data model is the "activity". This is an order submitted to the SUT by a user demanding the execution of a data processing operation. Examples: Compile a defined COBOL program; perform a defined data base request to an electronic personal information system; execute a compiled program using a defined input data set; open an interactive menu ; start one step of this menu by a mouse click.

Listing all activities of all users of the user entirety and classifying them yields the "activity types". An activity type is a class of activities which are regarded as equivalent. The equivalence is to a certain degree a discretionary decision of the designer of a workload. Examples: The activity type "compile medium size COBOL program" assumes compilations of COBOL programs having 500 to 2000 lines of code; the activity type "telephone number search" subsumes all database requests in the electronic telephone book of a company. Let w be the number of activity types of a defined workload and AT_i the name of the i-th activity type. Then the list of all activity types will be

$$AT_1,\ AT_2,\ AT_3,\ .....,\ AT_w\ .$$

The ISO data model assumes that a submission of an activity is preceded by a so-called preparation time, colloquial think time, which elapses before the task submission. Before starting the preparation time the user decides whether it begins with the submission of the preceding task or begins when the preceding task has been completed. The first case is the NOWAIT mode of task submission. The second case is the WAIT mode of task submission. In the ISO/IEC 14756 these modes are called "task modes".

Note: These two modes are not equivalent to the "dialog" (or "interactive") and "batch" mode in mainframe systems and UNIX systems. (See Sections 11.5.1.1, 11.5.2.1) for problems with "batch" modes.) ■

In other words: The task mode indicates whether the preparation time for the next task begins with the submission of the preceding activity or begins when the preceding activity has been completed.

In the ISO model, whenever a user submits a task he states his requirements concerning the execution time of the activity.

In the simplest case, this requirement is a time limit of the execution time ("threshold"). Example: Completion of the ordered activity within 5 seconds. But the requirement can also be a "worst case distribution" of the execution time. Simple example: 90% of the execution times may not be longer than 2 seconds; and no execution time may be longer than 10 seconds. These threshold and the worst case distribution are examples of so-called timeliness functions (see Section 2.5).

Let p be the total number of different timeliness functions used by the user entirety and TF_i the name of the i-th timeliness function. Then the list of all timely functions will be

$$TF_1,\ TF_2,\ TF_3,\ .....,\ TF_p\ .$$

Whenever a user orders the SUT to execute an activity he includes with it the value of the task mode and a timeliness function. Such a combination of specific activity, a specific value of the task mode and a specific timeliness function is called "task". Now we list all tasks which occur (when a defined user entirety is active) and classify them according to the triples of task type, task mode and timeliness function. I.e. we find out the so-called task types. Each task type is a triple

AT , M , TF

where AT is the activity type, M is the value of the task mode and TF is the timeliness function. We use (for simplicity) M=0 for NOWAIT and M=1 for WAIT. Having performed the above classification we find the set of different task types. The total number is named m. The j-th task type is named TT_j. The list of all task types will be

$$TT_1,\ TT_2,\ TT_3,\ .....,\ TT_m\ .$$

Note: The total number of possible task types is the product w*2*p. Usually, not all possible triples are used by the user entirety. Therefore it holds that $m \leq w*2*p$. ■

2.4 Explanation of the terms "chain" and "chain type"

The users submit, as explained in Section 2.2, typically sequences of tasks. A sequence of tasks is called task chain (abbreviated as "chain"). To describe a chain we have to list the sequence of the types of the tasks submitted by a user, as for instance

$$TT_3,\ TT_1,\ TT_4,\ TT_3\ .$$

Listing all different sequences yields the set of chain types. Let u be the total number of chain types and

$$CT_1,\ CT_2,\ CT_3,\ .....,\ CT_u$$

the list of all names of chain types. Example: u = 3.

$$CT_1 = TT_3,\ TT_1,\ TT_4,\ TT_3$$
$$CT_2 = TT_2$$
$$CT_3 = TT_1,\ TT_1$$

CT_2 shows an example of a task type which is intended to be executed "alone" and not included in preceding and following tasks.

The length of a chain type is the number of contained tasks. The length of CT_1 is 4, of CT_2 is 1 and of CT_3 is 2 .

2.5 Explanation of the timeliness function

A timeliness function (abbreviated as TF) is the description of the user requirements for the completion time of the ordered "activity" of the "task". The TF is specific to each task type.

Creating a timeliness function is shown by examples.

First example:
It starts with defining a basic requirement of the execution times. This is done by setting a time class limit $g_T(1)$ and the according relative time class frequency $r_T(1)$. For instance we set

$$g_T(1) = 2 \text{ seconds and } r_T(1) = 0.90 \quad .$$

This means that up to 90% of the execution times of the regarded task type may not exceed 2 seconds, and this setting allows that 10% of the execution times can be longer than 2 seconds. An upper limit, therefore, has to be set. For instance we define that no execution time may be longer than 10 seconds. Herewith we have stated the values of a second time class limit $g_T(2) = 10$ seconds and the according relative time frequency $r_T(2) = 1.0$. The created timeliness function has two classes. The total number of classes is named z . We have $z = 2$. Let the timeliness function be TF_1. TF_1 can be represented as shown in Fig. 2-4 .

TF_1	$z = 2$		
	k	$g_T(k)$	$r_T(k)$
	1	2.00 sec	0.90
	2	10.00 sec	1.00

Fig. 2-4 Example of a timeliness function having two time classes

Second Example:
We set the time class limit to 3 seconds. Additionally we state that no execution time may override this value. I.e. 100% of the execution times shall be between 0 and 3 seconds. Thus a "one class timeliness function" is defined. We have $z = 1$. Let the timeliness function be TF_2. TF_2 can be represented as shown in Fig. 2-5 .

TF_2	$z = 1$		
	k	$g_T(k)$	$r_T(k)$
	1	3.00 sec	1.00

Fig. 2-5 Example of a timeliness function having only one time class

Colloquially a "one class timeliness function" is called "threshold".

Third Example:
We set the basic requirement to be the same as in the first example, i. e. $g_T(1) = 2$ seconds and $r_T(1) = 0.90$. But with regard to the second time class we make an extension. 2% of the execution times may have a duration of up to 20 seconds. This defines a third time class. Let this timeliness function be TF_3. TF_3 can be represented as shown in Fig. 2-6 .

TF_3	z = 3		
	k	$g_T(k)$	$r_T(k)$
	1	2.00 sec	0.90
	2	10.00 sec	0.98
	3	20.00 sec	1.00

Fig. 2-6 Example of a timeliness function having three time classes

General rules:

- There is no general limit to the total number of time classes of a timeliness function. But in practice there are usually 2 or 3 classes.

- $g_T(k)$ and $r_T(k)$ must be increasing functions. I. e.:

$$0 < g_T(1) < g_T(2) < ..... < g_T(z) \qquad (2.1)$$

and

$$0 < r_T(1) < r_T(2) < ..... < r_T(z) \qquad (2.2)$$

- The term r_T of the uppermost class always has the value of one, i.e.:

$$r_T(z) = 1.00 \qquad (2.3)$$

2.6 The basic parameters of an ISO-type workload

Typically not all users of a user entirety are different. Often a user entirety contains users which have, with respect to the task types, the chains, the preparation times, the relative frequencies of usage of chain types etc. the same properties. Such users define a user type. Let n be the total number of different user types of a user entirety and $N_{user}(i)$ the total number of users of the i-th type. These values are basic parameters of the ISO workload. Additional basic parameters are shown in previous paragraphs. The complete set of all basic parameters is shown in Fig. 2-7.

The basic parameters are $1 + n + 1 + 1 + 1 + 1 = 5 + n$ values. The ISO workload consists of this set of user behaviour parameters (see Section 2.7), the application programs and their data (see Section 2.8) and the advanced parameters (see Section 2.9).

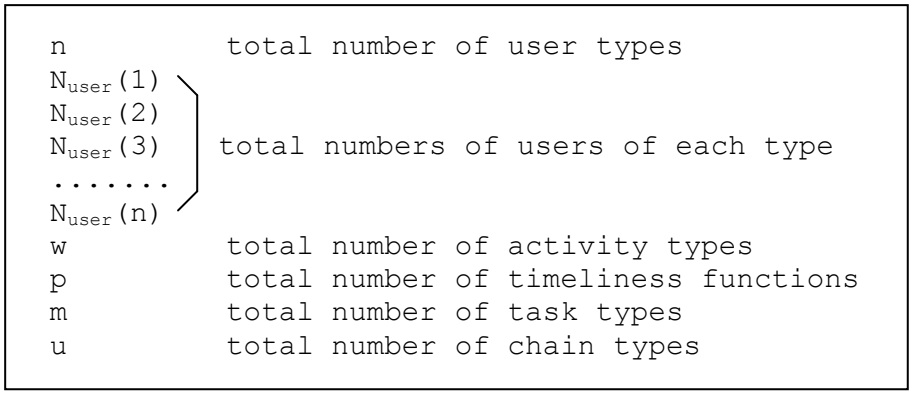


Fig. 2-7 The basic parameters of an ISO workload

The total number of users is N_{tot}.

$$N_{tot} = N_{user}(1) + N_{user}(2) + \ldots + N_{user}(n) \qquad (2.4)$$

A simple example of a basic parameter set is shown in Fig. 2-8 .

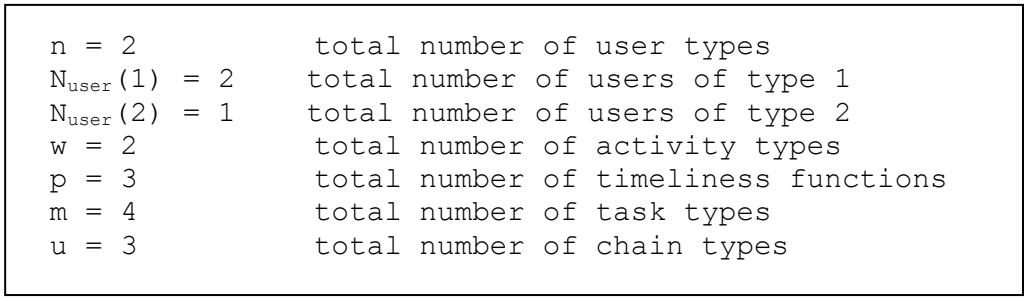


Fig. 2-8 Simple example of the basic parameter values of an ISO workload having $N_{tot} = 2 + 1 = 3$ users

2.7 The user behaviour parameters

The user behaviour is described using four lists and three matrices.
The four lists are:

- activity types
- task types
- timeliness functions
- chain types

The three matrices are:

- relative chain frequencies
- preparation time mean values
- standard deviation values of the preparation times

2.7.1 The activity type values

As shown in Section 2.3 an activity type represents a class of activities. One representative activity has to be chosen for the definition of the activity type. This activity is described by its input data which have to be submitted to the SUT. This is the usual situation, but there are two special cases.

Special case 1: The activity consists of several sequential steps, in which can be defined as one complex activity. The set of all partial input data is the input data of this activity. The sum of all partial preparation times is the preparation time of this activity. The execution time of this activity is the sum of all partial execution times.

Special case 2: Varying input data may be unavoidable. An example is with the activity "create a bank account". Whenever such an activity is submitted to the SUT a new bank account has to be created which has to be unique. This is an example of "input variation" within an activity type. The rules for varying the input data have to be specified for all activity types using input variation.

According to the total number w of activity types the list of activity types has w entries. Each entry consists of the following parts:

- Current number of the activity type.
- Logical meaning of the input.<
- Length of the input string (number of characters) submitted by the user or, in case of graphical input, the number of graphic actions (such as cursor movements and clicks).
- The input string itself or the list of graphical actions.
- Complete definition of all rules on how to modify the input data if there is any activity input variation.

Fig. 2-9 shows a simple example. The user entirety has only two activity types.

Name:	AT_1	AT_2
Activity type number:	1	2
Logical meaning of the input:	Name of a shell script	input to a program
length of the input string:	7 characters	6 characters
The input string itself:	testjob	showxx
Activity input variation:	- none -	yes *)

*) Rule: xx are two decimal digits which change randomly before each task submission to the SUT.

Fig. 2-9 Simple example of an activity type definition list (w = 2)

2.7.2 The task type values

As shown in Section 2.3 a task type is defined by a three values: Activity type, task mode and timeliness function. The task type list has m entries, each consisting of the following values

- Current number of the task type
- Number of the activity type used in the task type
- Value of the task mode M
- Number of the timeliness function stated for this task type

Fig. 2-10 shows a simple example of a user entirety with four task types.

Name	TT_1	TT_2	TT_3	TT_4
Task type number	1	2	3	4
Activity type number	2	1	1	2
Value of M	1	1	1	0
timeliness function number	2	1	2	3

Fig. 2-10 Simple example of a task type definition list (m = 4)

2.7.3 List of timeliness function values

The list has p entries, each consisting of:

- Current number of the timeliness function
- Number z of time classes of the timeliness function
- z pairs of values "time class limit g_T"
  and "maximum allowed relative frequency r_T"

Fig. 2-11 shows an example containing the three timeliness functions of Section 2.5 .

Name		TF_1	TF_2	TF_3
Current number		1	2	3
Number z of time classes		2	1	3
z couples	$g_T(1)$	2.00 sec	3.00 sec	2.00 sec
	$r_T(1)$	0.90	1.00	0.90
	$g_T(2)$	10.00 sec	-	10.00 sec
	$r_T(2)$	1.00	-	0.98
	$g_T(3)$	-	-	20.00 sec
	$r_T(3)$	-	-	1.00

Fig. 2-11 Example of a timeliness function definition list (p = 3)

2.7.4 List of chain type definitions

The list has u entries as explained in Section 2.4 . Each entry consists of:

- Name of the chain type
- Current number of the chain type
- Length of the chain (total number of tasks)
- The sequence of task types

Fig. 2-12 shows an example containing the three chain types of Section 2.4 . For "length of chain type" (number of contained tasks) see Section 2.4 .

Name	CT_1	CT_2	CT_3
Current number	1	2	3
Length	4	1	2
task type sequence	3,1,4,3	2	1,1

Fig. 2-12 Example of a chain type definition list (u = 3)

2.7.5 The relative chain frequencies

In this section we assume at first a simple situation of only one user type. Each user uses the three chain types which were defined in Section 2.7.4 . As explained in Section 2.2 the relative frequencies of submitting chains have to be defined. For instance the users of user

type 1 submit 10% chains of the chain type 1, 50% of chain type 2 and the rest (i.e. 40%) of chain type 3. This yields the list shown in Fig. 2-13a .

chain type	relative frequency
1	0.10
2	0.50
3	0.40

Fig. 2-13a Relative chain frequencies of users of user type 1

According to the example of Fig. 2-8 we now introduce a second user type. We assume that the users of this type submit 30% chains of chain type 1, no chains of chain type 2 and the rest (i. e. 70%) of chain type 3. This yields the list shown in Fig 2-13b .

chain type	relative frequency
1	0.30
2	0.00
3	0.70

Fig. 2-13b Relative chain frequencies of users of user type 2

In the ISO standard the relative chain frequencies are represented by the term q(i,*l*) where i is the current number of the user type and *l* is the current number of the chain type. Putting together the q−lists for all user types yields the so-called q−matrix which has u rows and n columns. The q−matrix resulting from Figures 2-13a and 2-13b is shown in Fig. 2-14 .

l= \ i=	1	2
1	0.10	0.30
2	0.50	0.00
3	0.40	0.70

q(i,*l*)

q-matrix with

q(1,1) = 0.10, q(1,2) = 0.50, q(1,3) = 0.40

q(2,1) = 0.30, q(2,2) = 0.00, q(2,3) = 0.70

Fig. 2-14 Example of a q-matrix (u = 3 chain types, n = 2 user types)

Please note firstly that a q−value never can be greater one; and secondly that the sum of all values of each column has to equal exactly one.

2.7.6 Preparation time mean values

Section 2.3 explained that each submission of a task is preceded by a user preparation time (think time).

The preparation time varies randomly if this user subsequently submits tasks of this type. The mean value of the preparation times preceding the tasks of the j-th task type is h. In our example there are m = 4 task types. Consequently we have to define four h-values as, for example, in Fig. 2-15a .

task type	preparation time mean value
1	10.0 sec
2	3.0 sec
3	15.0 sec
4	1.0 min

Fig. 2-15a Preparation time mean values of users of user type 1

In our example we have two user types. It may be that user type 2 uses preparation times having mean values which are different from those of user type 1. The values listed in Fig. 2-15b are an example.

task type	preparation time mean value
1	2.0 sec
2	2.0 sec
3	10.0 sec
4	30.0 sec

Fig. 2-15b Preparation time mean values of users of user type 2

In the ISO standard h(i,j) is the term for the preparation time mean values where i is the current number of the user type and j is the current number of the task type.

Putting together the h-lists for all user types yields the so-called h-matrix. It has m rows and n columns. The h-matrix resulting from Figures 2-15a and 2-15b is shown in Fig. 2-16 .

i= 1 2

j=		
1	10.0 sec	2.0 sec
2	3.0 sec	2.0 sec
3	15.0 sec	1.0 sec
4	60.0 sec	30.0 sec

h(i,j)

h-matrix with

h(1,1) = 10.0, h(1,2) = 3.0, h(1,3) = 15.0, h(1,4) = 60.0
h(2,1) = 2.0, h(2,2) = 2.0, h(2,3) = 1.0, h(2,4) = 30.0

Fig. 2-16 Example of a h-matrix
(m = 4 task types, n = 2 user types, values are in seconds)

2.7.7 Preparation time standard deviation values

The standard deviation s corresponding to each mean value has to be defined. This yields the s-matrix. It has, like the h-matrix, m rows and n columns. s(i,j) is the standard deviation of the preparation times of a user of the i-th type before submitting a task of the j-th type. An example of an s-matrix is shown in Fig. 2-17 .

i= 1 2

j=		
1	2.0 sec	0.1 sec
2	0.5 sec	0.2 sec
3	0.0 sec	0.2 sec
4	10.0 sec	8.0 sec

s(i,j)

s-matrix with

s(1,1) = 2.0, s(1,2) = 0.5, s(1,3) = 0.0, s(1,4) = 10.0
s(2,1) = 0.1, s(2,2) = 0.2, s(2,3) = 0.2, s(2,4) = 8.0

Fig. 2-17 Example of a s-matrix
(m = 4 task types, n = 2 user types, values are in seconds)

Notes:

1. A `s`-value being zero implies that the preparation time does not change randomly . It always has the mean value which is defined in the `h`-matrix. ■

2. A `s`-value may not be too large with respect to the corresponding mean value. A rule of thumb is that the `s`-value should not exceed 50% of the `h`-value. Otherwise in an example of a unique distribution negative preparation times would appear. Negative times cannot occur in reality. For this problem see also Section 6.3.3 . ■

2.8 Application programs, their data and computational results

All programs needed for the execution of the set of task types have to be presented on a digital storage medium (either as executable programs or as the complete source code). These programs have to be ready for use on the SUT. The required operating system command procedures (OSCP) also have to be included. The OSCP are of exceptional importance if the input string of one or more activity types refer to the name of such an OSCP.

All data which are needed by the application programs and OSCPs for their operation have to be presented on a digital storage medium. The amount of such data can be small (for instance only some small files). But it can also be huge (for instance the content of a large data base).

For all task types the correct computational results (including possible changes of stored data) have to be specified and listed on a digital storage medium in order to be available for verifying the correct operation of the SUT. If there is any activity type input variation then all modified computational results and variations of task output have to be listed corresponding to the rules of input variation. Alternatively, the rules of output variation can be defined. It is essential that these rules are comprehensive and complete.

2.9 The advanced parameters of an ISO-type workload

The ISO standard requires three classes of validation.

2.9.1 Computational results

The first validation checks the computational results of the SUT. Each activity must have been completed with these results. The validation is done with reference to the "correct results" recorded as explained in Section 2.8 .

2.9.2 Precise working of the RTE

The second validation checks that the RTE created user behaviour values sufficiently near those specified in Section 2.7 . (See also Section 4.2). There are three check criteria.

$DELTA_q$: The maximum acceptable relative difference of the measured chain frequencies compared with those in the q-matrix.

$DELTA_h$: The maximum acceptable relative difference of the measured mean preparation times compared with that in the h-matrix.

$DELTA_s$: The maximum acceptable relative difference of the measured preparation time standard deviations compared with that in the s-matrix.

Note: The ISO standard allows DELTA values which are different for each combination of user type and a second argument (chain type or task type). But usually a single DELTA values is used (see Sections 4.2.2 to 4.2.4) . ■

The values of the three DELTA terms have to be defined. Good values of the first two terms are

$$0 \leq DELTA_q \leq 0.01 \qquad (2.5)$$

$$0 \leq DELTA_h \leq 0.02 \qquad (2.6)$$

If DELTA limits greater those of (2.5) or (2.6) are used, a less satisfactory accuracy of the RTE will be tolerated. This implies serious errors in the measured performance values.

A greater $DELTA_s$ limit is usually acceptable. A value of 0.05 is very good; but 0.20 is sufficient. Therefore $DELTA_s$ should be set as

$$0 \leq DELTA_s \leq 0.20 \qquad (2.7)$$

2.9.3 Statistical significance

The third validation checks the statistical significance of the measured performance values (see Section 4.3). There are two criteria (referring to a so-called sequential statistical test):

- The confidence coefficient ALPHA of the mean execution times
- m confidence intervals "2*d(j)" of the mean execution times
  (j is the current number of the task type)

In practice ALPHA = 0.05 is a very good value. It gives a very high significance but yields typically a long measurement duration. ALPHA = 0.10 is usually sufficient. But ALPHA should not be set greater than 0.20 .

With respect to d(j), practical experience is that all d(j) values can have the same relative value (with respect to the mean execution time). I.e. in practice

```
d = 0.05 * (mean execution time)
```

is a very strong value which yields typically a long measurement duration. But

```
d = (0.10 ... 0.20) * (mean execution time)
```

is mostly sufficient.

Setting

```
d = 0.3 * (mean execution time)
```

yields a poor statistical significance.

The quotient

$$d_{rel} = d/(\text{mean execution time}) \qquad (2.8)$$

is called relative half width confidence interval. This value is often used instead of `d` .

2.10 Short summary of the contents of an ISO workload

The workload consists of

- basic parameters, see Section 2.6,
  and user behaviour parameters, see Sections 2.7.1 to 2.7.7
- application programs (programs, OSCPs,...) and data
  (input data, stored data, correct computational results), see Section 2.8
- advanced parameters
  (`DELTA` values and significance values), see Section 2.9

To become more familiar with the contents and meanings of the workload description, perform the exercises in Section 2.11. The detailed format of the workload description is defined in the Normative Annex C of ISO/IEC 14756. The set of parameters in Sections 2.6 and 2.7 of this book is called "workload parameter set" (WPS) in the standard. For the ISO representation of a workload in a directory see Sect. 11.1 of this book.

2.11 Exercises

Exercise 1: ISO format of the WPS

Write down the workload parameter as is defined by the examples in Sections 2.6 and 2.7 .

Exercise 2: A very simple PC workload

Define an ISO-type workload generated by a single PC user. The user is assumed to execute in the DOS shell the two simple DOS commands "DIR" and "print a file containing a page of alphanumeric ASCII text". Complete the missing data with values of your own choice. Write down the WPS in ISO format.

Exercise 3: A very simple UNIX workload

Rewrite the workload defined in Exercise 1 for a UNIX system having only one active user. The user corresponds with the machine via a normal UNIX shell.

Exercise 4: A multi-user workload

Define an ISO-type workload generated by two user groups using a UNIX system via a normal command shell. The first user group consists of 3 users as in Exercise 3. The second group consists of one user performing program tests. He compiles a short C program (assumed to have no syntax errors). Then he runs the compiled program twice using different input data sets (assumed to produce no run time errors). The user runs this sequence repeatedly. Complete missing data with values of your own choice.

Exercise 5: Examples of the WLP input format of the DEMO system

Write down the WLPs of the workloads of Exercises 1, 2 and 4 in the input format of the DEMO system. Then perform for each of these WLPs a dummy run of the DEMO system only reading the WLP file and printing out its values. Check if they match the ISO WLP.

Solutions

For solutions see file

```
CD/Solutions/Solutions-Section2-11.pdf .
```

3. The measurement experiment

3.1 Principles of operation of the ISO-type user emulator (RTE)

The user emulator RTE emulates the set of users as defined in the basic parameter of the WLP. In the example of Section 2.6, Fig. 2-8 the RTE has to emulate two users of user type 1 and one user of user type 2. All users have to be emulated independently. The principle of operation of emulating a user will be demonstrated by user number 2 of user type 1.

The user chooses randomly a chain type e.g. CT_1 . The task type sequence is 3,1,4,3 (see Fig. 2-12 of Section 2.7.4). He submits 4 tasks in this sequence. Details of this submission will be shown later. After the submission of the 4th task the user chooses randomly a chain type, for instance chain type 3. The task sequence is 1,1 (see again Fig. 2-12). The user submits a task of task type 1, followed by another task of task type 1 . After this submission the user chooses randomly a chain. e.g. it is of chain type 2 which yields the submission of only one task (of task type 2, see Fig. 2-12). And so on.

The decision of the chain type has to be done randomly but according to the preset q-values of the WPS (see Section 2.7.5, Figures 2-13a and 2-14). The values in our example are

10%, 50%, 40% for the chain types 1, 2, 3 .

Later it will be shown how to implement a suitable random generator, see Section 3.6.1 and also the "Urn Method" (described in Chapter 6).

It is assumed that a random sequence of 10 chains is to be generated, consisting of the chain types 1, 2 and 3, where (according to the relative chain frequencies cited above) chain type 1 is included once, chain type 2 is included 5 times and chain type 3 is included 4 times. An example of a (random) sequence is as follows:

| 1 | 3 | 2 | 3 | 2 | 3 | 2 | 2 | 3 | 2 |

The 10 chains yield 17 tasks (according to Fig. 2-12 in Section 2.7.4) tasks. The resulting sequence of task types is as follows:

| 3,1,4,3 | 1,1 | 2 | 1,1 | 2 | 1,1 | 2 | 2 | 1,1 | 2 |

Before submitting a task the user emulator waits for a "task preparation time" , colloquially called think time. This preparation time has to be randomly chosen as follows. The preparation time preceding task type 1 is named h1 . Those of task type 2 is named h2 etc. Inserting this terms into the task type sequence yields the following sequence of pairs of preparation time and following task type number.

| h3 3, h1 1, h4 4, h3 3 | h1 1, h1 1 | h2 2 | h1 1, h1 1 |
| h2 2 | h1 1, h1 1 | h2 2 | h2 2 | h1 1, h1 1 | h2 2 |

h1 appears 9 times and has to be chosen randomly. But the h1 must have a mean value of 10.0 seconds (according to Section 2.7.6, Figures 2-15a and 2-16) and a standard deviation of 2.0 seconds (according to Section 2.7.7, Fig. 2-17). How to generate such a series of random values will be shown later (see Section 3.6.2; see also the "Urn Method" described in Chapter 6). Let be the name of the 9 values h11 to h19 .

h2 appears 5 times (mean time 3.0 seconds, standard deviation 0.5 seconds). Let be h21 to h25 the names of the 5 values.

h3 appears twice (mean time 15.0 seconds). Let the name of the two values be h31 and h32 . The standard deviation is zero (see Section 2.7.7, Fig. 2-17). This means that both h3 values are the same and are equal to the mean value of 15.0 seconds.

h4 appears only once. Let be h41 the name of this value. This single value must be equal the defined mean value of 60 seconds. The stated standard deviation is 10.0 seconds yielding the following problem.

A random variable appearing only once has a standard deviation of zero. Therefore the chosen task sequence cannot fulfil the required value of the standard deviation of the preparation time preceding the task type 4 which was set to be 10.0 seconds. This example shows a situation which often arises. The chain sequence is too short. It must be long enough so that every activity type, with a non-zero standard deviation appears at least twice. The solution is simple. Use a double chain sequence (i.e. a sufficiently long measurement duration). Then the problematic task type appears twice and the stated standard deviation can be realised. We ignore this problem for the moment and continue using the above sequence.

We replace the hx terms by the hxy terms, where x is the task type and y is the current task number of this type, and get the following sequence.

```
| h31 3, h11 1, h41 4, h32 3 | h12 1, h13 1 | h21 2 |
| h14 1, h15 1 | h22 2 | h16 1, h17 1 | h23 2 | h24 2 |
| h18 1, h19 1 | h25 2 |
```

Before submitting a task to the SUT the user takes the task type definition (see Fig. 2-10 in Section 2.7.2) to determine the activity type of the task. Additionally he gets the value of the task mode M. M indicates if the think time starts immediately after the submission of the preceding task (M = 0) or if the user has to wait for the completion of the preceding task (M = 1). For the M-values in our example see Fig. 2-10 . Inserting the task mode values (0 or 1) on the left of each hxy yields the following sequence.

```
|1 h31 3, 1 h11 1, 0 h41 4, 1 h32 3 |1 h12 1, 1 h13 1 |1 h21 2 |
|1 h14 1, 1 h15 1 |1 h22 2 |1 h16 1, 1 h17 1 |1 h23 2 |1 h24 2 |
|1 h18 1, 1 h19 1 |1 h25 2 |
```

Replacing the task type numbers by the activity type numbers (with reference to Fig. 2-10) yields the following sequence.

```
|1 h31 1, 1 h11 2, 0 h41 2, 1 h32 1 |1 h12 2, 1 h13 2 |1 h21 1 |
|1 h14 2, 1 h15 2 |1 h22 1 |1 h16 2, 1 h17 2 |1 h23 1 |1 h24 1 |
|1 h18 2, 1 h19 2 |1 h25 1 |
```

This sequence is not yet complete. The `hxy` values still have to be inserted. An example of suitable random values in seconds is the following list. They are computed by the Urn Method (see Section 6.3.2) and have the mean values and standard deviations as set above.

```
h11 = 09.225   h21 = 02.293   h31 = 15.000   h41 = 60.000
h12 = 11.549   h22 = 03.354   h32 = 15.000
h13 = 07.676   h23 = 03.707
h14 = 13.098   h24 = 03.000
h15 = 06.902   h25 = 02.646
h16 = 10.000
h17 = 12.324
h18 = 10.775
h19 = 08.451
```

Inserting these values yields the sequence shown in Fig. 3-1 . Because the sequence represent 17 tasks, it has 17 triples, each consisting of

- value of the task mode (0 or 1)
- preparation time (in seconds)
- activity type number

```
|1 15.000 1, 1 09.225 2, 0 60.000 2, 1 15.000 1 |1 11.594 2, 1 07.676 2|
|1 02.293 1 |1 13.098 2, 1 06.902 2 |1 03.354 1 |1 10.000 2, 1 12.324 2|
|1 03.707 1 |1 03.000 1 |1 10.775 2, 1 08.451 2 |1 02.646 1|
```

Fig. 3-1 Example sequence of 17 tasks each described by a triple of (task mode value, preparation time, activity type number)

The sequence shows the steps which the RTE has to execute when emulating the user in our example. The sequence contains one example without waiting for the result of the preceding task. The preparation time of the third task has to begin immediately after the submission of the second task. The preparation times of all the other tasks have to begin only after the SUT has completed the preceding task.

Finally the activity type numbers should be replaced by the corresponding input strings (see Fig. 2-9 in Section 2.7.1) which have to be submitted to the SUT for starting the task execution. In case of type 2 activities the last two characters of the strings have to be modified according to the rule described in Fig. 2-9 . This yields the detailed list of steps which are to be performed by the emulator of the second user of the first user type in our example. It is the task list of this user. Analogously can the task lists be set up for the first user of the first user type and the only one user of the second user type.

In our example the users interact with the SUT via alphanumerical commands. If they could interact using graphic interactions only a slight change would be necessary. The activity type numbers do not have to be replaced by the alphanumerical input strings but by the graphical action control data such as the mouse clicks or cursor positioning data.

For each user an individual sequence of the type shown in Fig. 3-1 has to be created and thereof a task list. Each user has to be to be emulated independently from the others. Each individual user emulator has to record a detailed logfile any which has to contain all actions, time stamps for every step and any additional information (see Section 3.4). Also the computational results have to be recorded, mostly not in the logfile but in a separate file (see Section. 3.5).

3.2 Dynamic task generation versus pregenerated task lists

An obvious way to realise a user emulator is the "dynamic" one. This means that the emulator randomly chooses a chain type, defines the step elements of the chain and executes it. Then the emulator randomly chooses the next chain type etc. Many references to all previous task data have to be checked in order to retain the relative chain frequencies, the preparation time mean values and the required standard deviations. Additionally the input string variation of the relevant activity types and many other operations have to be performed. Very fast hardware is needed for the emulation of a user. Typically if the total number of users N_{tot} (see Section 2.6, formula (2.4)) is large then a vast amount of hardware is needed to realise the RTE which has to work in realtime and without time delays.

This amount can be reduced by pregeneration of the task lists. The idea is simple. A program generates a task list for each of the N_{tot} users. The generation has to be performed according to Section 3.1 . The lists have to be long enough to realise a suitable duration of the measurement. The pregeneration has to be done separately before the measurement run. Therefore it is not time critical. It is not necessary to generate the lists simultaneously. The generation can be performed separately for each user. When completed the N_{tot} emulators will be started simultaneously. The work of an emulator, which only has to execute the steps of a pregenerated list needs significantly less hardware speed than the dynamic task generation. What is needed is only a task submitter which acts according to the steps in the list. It is possible to emulate even a large number of users by a typical medium size computer.

The actions of such a task submitter are simple.

Step 1: Read an entry (task element) from the task list.
(It contains the task mode value, the preparation time value and the input string.)
Step 2: Wait for the task completion of the preceding task if the task mode value is 1. Otherwise do not wait.
Step 3: Wait according to the preparation time value.
Step 4: Submit the input string to the SUT. Then go to step 1.

Note: At the beginning the emulator has to neglect the position "task mode" of the very first entry, due to fact that there is no preceding task. ■

There is an important fact to be considered. Present day operating systems are typically not able to perform the actions "wait for the result of the preceding task before submitting the actual task" and "do not wait to the result of the preceding task before submitting the actual task". What the operating systems can do is to bring a submitted task either into the foreground, "interactive job", or into the background, "batch job". Therefore the submitter

cannot interpret the task mode values "0" and "1" for "batch" and "interactive" or vice versa. But there is a solution. The task submitter has to perform a "look-ahead" as follows. When processing a task element the task submitter also has to check the next task element. If the task mode of this next task element is "0" then submit the actual task as a background job. If the task mode of this next task element is "1" then submit the actual task as an interactive job. The task submitter has to perform dynamically a so-called look-ahead before processing a task element .

The necessity for implementing the look-ahead operation in the submitter can be avoided by a modification of the task list as follows. Replace the task mode value by "B" if the task mode value of the next task element is "0". Replace the task mode value by "I" if the task mode value of the next task element is "1". Fig. 3-2 shows the list of Fig. 3-1 after performing this modification.

```
|I 15.000 1, B 09.225 2, I 60.000 2, I 15.000 1 |I 11.594 2, I 07.676 2|
|I 02.293 1 |I 13.098 2, I 06.902 2 |I 03.354 1 |I 10.000 2, I 12.324 2|
|I 03.707 1 |I 03.000 1 |I 10.775 2, I 08.451 2 |I 02.646 1|


   Sequence of 17 tasks each described by a triple of
   (batch/interactive mode, preparation time, activity type number)
```

Fig. 3-2 Example sequence of Fig. 3-1 after look-ahead transformation.

The operation principle of the task submitter for such a list is easier to implement with respect to present day operating systems. The principle is as follows:

Step 1: Read entry (task element) from the task list.
(It contains the batch/interactive mode, the preparation time value and the input string.)
Step 2: Wait according to the think time value.
Step 3: Submit input string for a background job if the mode is "B"; submit it for an interactive job if the mode is "I".
Step 4: Wait until the SUT is ready for receiving the next job submission. Then go to step 1.

Note: The mode, background or interactive, of the last task element in the list can be ignored. This is due to fact that there is no following task. ■

Reuse of pregenerated task lists: Naturally, a pregenerated task list can be reused for a new measurement. There is no technical reason to make this impossible. However, this violates the idea of the ISO standard which states that tasks shall be randomly chosen. The danger is that people could make efforts of tuning the SUT for one defined pregenerated task list. This is not permitted in the standard. After the measurement one should delete this list and generate a new list when repeating the measurement of the same SUT as well as when measuring a different SUT.

3.3 The three phases of a measurement run

The ISO measurement procedure consists of three phases as shown in Fig. 3-3 :

- stabilisation phase (StP)
- rating interval (RI)
- supplementary run (SR)

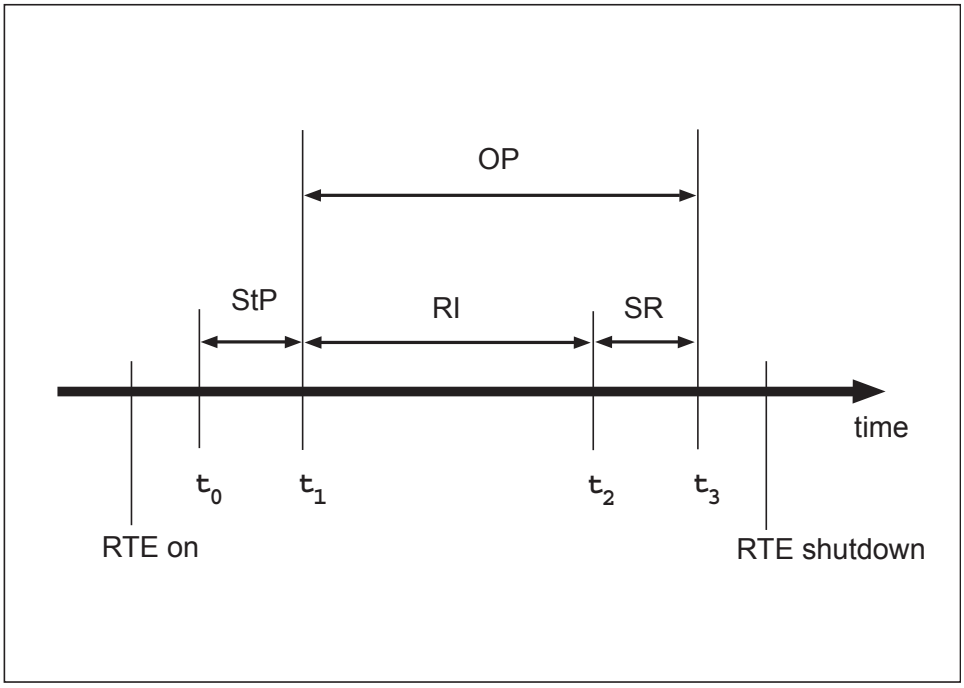


Fig. 3-3 The three phases of a measurement run

The StP: The RTE is activated on. After some time it starts to work. This is time t_0. The login procedures of the emulated users are either started at once or after arbitrary times. The StP is intended to bring the SUT into a stable state of operation. There is no regulation in the ISO standard concerning the duration of the StP. The measurement operator decides on the duration according to his experience. After the logins, the RTE has to start the task submissions according to the workload parameter values. The completion times of the tasks are not taken into account for the computation of the performance values. The end of the StP is the time t_1 .

The RI: The RI begins at t_1 . The task submission has to be performed according to the workload parameter values. The completion time of each task submitted during the RI is taken into account for the computation of the performance values. The RI duration has to be chosen appropriate to the workload. If the duration is too short then at least two problems may arise. First there is the risk that the check criteria of statistical significance of the measured performance values will not be satisfied. Second there is the risk that the check criteria of sufficient precise work of the RTE will also not be satisfied. (For these two aspects see Section 2.9 .) The end of the RI is t_2 .

The SR: According to the ISO requirements the RTE should not be stopped at the end of the RI. It must continue to work according to the workload parameters until time t_3 . This is the moment when all tasks submitted within the RI are completed. The SR is needed to ensure an identical statistical load situation for uncompleted tasks in the RI. If the RTE were stopped at t_2 then those tasks which were still running at t_2 would be completed earlier due to less load on the SUT. The RTE may be shutdown either at t_3 or later.

The OP: The time from t_1 to t_3 is the observation period (OP). For this period the measurement procedure has to be validated by checking all tasks submitted to the SUT (for details see Chapter 4).

Note (concerning the duration of the StP): The longer the StP the better. But if the StP is longer than necessary then there is no additional advantage. If the StP is too short then the load may fluctuate too much in the first part of the RI; i. e. the RI would have to be lengthened for achieving the required statistical significance of the performance values. ■

3.4 The logfile (measurement result file)

The historical data of all tasks submitted during the OP have to be recorded in a logfile. For each task a data set, "logfile record", has to be created. According to "Annex D" of ISO/IEC 14756, each logfile record has to contain at least the following entries.

- Current number of the task listed chronological since t_1, regardless of the user submitting the task.
- Type number of the user submitting the task.
- An identification of the user submitting the task (e.g. a string or the current number of the user within the group of users of the same type).
- Type number of the task. Additional the actual submitted input string of the activity is recommended.
- Type number of the chain containing the task.
- Sequential number of the task within the chain.
- Time stamp at the start of the preparation time preceding the task.
- Time stamp at the task submission.
- Time stamp at the completion of the task.

The ISO standard proposes that a typical time stamp has a precision of 1/100 second. But it recommends that it be modified to a proper value (greater or smaller) if needed.

With respect to the time stamps the ISO standard defines the following:

- task submission (begin of execution time) is the event when the total input (strings, files, cursor movements, etc.) is completely received at their destination in the SUT.

- task completion (end of execution time) is the event when the total output (strings, files, graphic information, etc) is completely received at their destination.

Distinction is needed between the external and internal task results. The external task result is the information seen by the user at his interface (usually a computer screen). The internal task result is the information stored on the hardware, such as in files and in databases. The internal task result is not explicitly shown to the user submitting the task. The event of task completion occurs when both the external and internal task results are completed and received at their destinations.

Of course the logfile need not to be confined to one physical file; it could be a set of files or parts of a data base system. It is recommended that the logfile be divided logically into two parts: an RI part and an SR part. This makes it easier to perform the validation (see Chapter 4).

3.5 Storing the computational results

All computational results produced by the tasks running on the SUT have to be stored during the observation period (possibly shortened or compressed). This computational result file has to be complete enough to prove unequivocally the correct execution of all tasks processed by the SUT and their correct computations.

As with the logfile, the computational result file need not be confined to one physical file. It is recommended that this also be divided logically into two parts: an RI part and an SR part. This makes it easier to perform the validation (see Chapter 4).

For dynamic correctness checking without storing computational results see Section 4.4 .

3.6 Some random generation methods

3.6.1 Generation of random chain type numbers

The relative chain type frequencies required by users of a defined type are specified in the q-matrix of the WPS. For instance, the relative chain frequencies of a user of user type 1 in our example (see Fig. 2-14 in section 2.7.5) are as follows:

chain type	relative frequency
1	0.1 (i.e. 10%)
2	0.5 (i.e. 50%)
3	0.4 (i.e. 40%)

To generate a random sequence of chain type numbers having this relative frequencies we use a unique distributed random generator. It is available as a standard function in most programming languages. Such a generator produces random numbers in the interval (0,1).

To generate the chain type numbers we divide the interval (0,1) into subintervals according to the chain frequencies. In our example we get the intervals

(0.0, 0.1) for chain type 1 ,
(0.1, 0.6) for chain type 2 ,
(0.6, 1.0) for chain type 3 .

To obtain a chain type number the random generator is started. It could deliver the value 0.155, which is included in the second interval, yielding chain type 2. The next random number could be 0.720, yielding chain type 3. And so on. Repeating this several times yields a random sequence of chain type numbers approximating to the required distribution. The approximation is improved by lengthening the sequence, (see also Section 3.6.3). See Chapter 6 for the "Urn Method" which does not only approximate but delivers an exact distribution with a finite length of the sequence.

3.6.2 Generation of random preparation times

To generate random preparation times we use the fact that the distribution type of a random variable $\underline{X} = c*\underline{x}$, where c is a constant, is the same as of $\underline{x}$.

Example: Preparation time values of the task type 4 for user type 2
(according to Section 2.7.6, Fig 2-16):
The required mean time is 30.0 seconds. We use the same unique distributed random generator as in Section 3.6.1 . We start it and it produces, for instance, the value 0.177 . Then the preparation time value will be 0.177*(2*30.000) = 10.620 seconds. Next random number might be 0.899. Then the preparation time value would be
0.899*(2*30.000) = 53.94 seconds. And so on.

Note: The factor of 2 in the formula is necessary because the mean value of a "(0,1) unique distributed" random variable equals 0.5 and not 1.0 . ■

Such a generated preparation time sequence is an approximation. It is improved by lengthening the sequence. For this problem see Section 3.6.3 .

The simple method of the example cannot alter the value of the standard deviation. The longer the sequence, the nearer the standard deviation will approach that of the
"(0 , 60.00) unique distribution". For more details see specialist literature of mathematical statistics. There are other methods described for generating sequences of general distribution types having a predefined standard deviation value. Additionally compare Section 3.6.3 for problems with finite random sequences.

For a method which does not only approximate but deliver a distribution keeping exact mean value and standard deviation even in case of a finite length of the sequence see Chapter 6 ("Urn Method").

3.6.3 A practical problem with finite random sequences

A real measurement has a defined duration. A defined number of task chains and of preparation times occur. Contrary to this, the random generation methods as discussed above presume an "infinite series of samples". Therefore the relative frequencies of a finite long random chain type sequence differ from the required values. The mean values and standard deviations of a finite number of preparation times also differ from the specified values. The values are improved by lengthening the sequence, i.e. by lengthening the measurement duration. As a consequence we have to take into account that the preparation time mean values, their standard deviations and the relative chain frequencies of an RTE differ from the specified values. (These values are `q-`, `h-` and `s-`values of the WPS.) This fact is the reason for introducing the `DELTA` validation criteria (see Section 2.9.2).

3.7 Exercises

Prerequisite: Provide a random generator producing unique distributed numbers in the interval (0,1). This can for instance easily be realised by a pocket calculator having a random function or by writing a little program using a standard subroutine "RANDOM".

Exercise 1: Inaccuracy of the mean values

The exercise 1 assumes that the mean preparation time of a defined task is set to 10.0 seconds.

Part 1
Generate, using the method of Section 3.6.2 three sequences each having 5 random preparation times.

- Compute the mean value of each sequence.
- Compute the relative differences with the required mean of 10.0 seconds.
- Compare these relative differences.

Part 2
Generate, using the same method, three similar sequences each having 10 random preparation times.

- Compute the mean time of each sequence.
- Compute the relative differences to the required mean of 10.0 seconds.
- Compare these relative differences with those of Part 1. Rate the improvement (if any) in the approximations of the required mean compared with those of Part 1 .

Part 3
Repeat Part 2 with each sequence having 20 random preparation times. Additionally, rate the improvement (if any) in the approximations of the required mean compared with those of Parts 1 and 2.

Exercise 2: Generation of task lists

Use the following WPS.

```
# File: ch3-exercise2-wps.txt

       Workload parameter set
       ======================

1.     Basic parameter values
------------------------------

       (1) Total number of different user types: n = 2 ;

       (2) Total amount of emulated users of each type: N_user(1) = 1
                                                        N_user(2) = 1

       (3) Total number of different activity types: w = 3 ;

       (4) Total number of different timeliness functions: p = 1 ;

       (5) Total number of different task types: m = 3 ;

       (6) Total number of different chain types: u = 3 ;


2.     Activity type definitions
---------------------------------


       Activity type number:   |   1    |    2    |    3    |
       ------------------------|--------|---------|---------|
                               |        |         |         |
       (1) The logical meaning |        |         |         |
           of the input:       |   *)   |      *) |    *)   |
                               |        |         |         |
       (2) The length (number  |        |         |         |
           of characters) of   |        |         |         |
           the input string:   |   3    |    3    |    3    |
                               |        |         |         |
       (3) The input string    |        |         |         |
           itself:             |  TT1s  |  TT2s   |  TT3s   |
                               |        |         |         |
       (4) Activity type       |        |         |         |
           input variation:    |  none  |  none   |  none   |

       *)  The name of a shell script


3.     Task type definitions
-----------------------------


       (1) Current number j of the task type:        |  1  |  2  |  3  |
       ----------------------------------------------|-----|-----|-----|
                                                     |     |     |     |
       (2) Number of the activity type:              |  1  |  2  |  3  |
                                                     |     |     |     |
       (3) Value of the task mode M(j): :            |  1  |  1  |  1  |
                                                     |     |     |     |
       (4) Type number of the timeliness function:   |  1  |  1  |  1  |
```

```
4.      Definitions of the timeliness functions (TF)
-----------------------------------------------------


        (1) Order number of the TF:             |    1    |
        --------------------------- ----------|---------|
                                                |         |
        (2) Number of time classes:             |   z=2   |
                                                |         |
        (3) z couples of values g_t and r_t,    |         |
            where g_t is the time limit and     |         |
            r_t is the maximum accepted         |         |
            relative frequency:         g_t(1):| 1.0 sec |
                                        r_t(1):| 0.80    |
                                               |         |
                                        g_t(2):| 2.0 sec |
                                        r_t(2):| 1.00    |
                                               |         |


5.      Definitions of chain types
----------------------------------


        (1) The current number l of the chain type:   |  1  |  2  |
        ---------------------------------------------|-----|-----|
                                                     |     |     |
        (2) The length L_chain(l) of the chain:      |  2  |  2  |
                                                     |     |     |
        (3) The sequence of the task type numbers:   | 1,2 | 2,3 |



6.      Definition of the chain probabilities
---------------------------------------------


          l | q(1,l) | q(2,l) |    l = current number of chain type
         ---|--------|--------|
          1 |  0.80  |  0.20  |
            |        |        |
          2 |  0.20  |  0.80  |
            |        |        |


7.      Preparation time mean values
------------------------------------


         j | h(1,j) | h(2,j) |       j = current number of task type
        ---|--------|--------|
         1 | 10 sec | 10sec  |
           |        |        |
         2 | 10 sec | 10sec  |
           |        |        |
         3 | 10 sec | 10 sec |
           |        |        |


8.      Preparation time standard deviations
--------------------------------------------


         j | s(1,j) | s(2,j) |       j = current number of task type
        ---|--------|--------|
         1 | 3 sec  | 3 sec  |
           |        |        |
         2 | 3 sec  | 3 sec  |
           |        |        |
         3 | 3 sec  | 3 sec  |
           |        |        |

# == End of WPS ==
```

This WPS is found in file

CD/Sol-files/wps-of-exercises/ch3-exercises2.txt .

Write down the sequence of tasks for all users of this WPS. The length of the sequences should be 5 chains for each user. Generate the chain sequences using the method described in Section 3.6.1 . Generate the preparation times using the method described in Section 3.6.2 .

Exercise 3: Chain sequence duration

Compute - for the sequences generated in Exercise 2 - the expected duration of the sequence of each user. For computing these values assume the following mean execution times of tasks:

Mean completion of TT_1 = 2.0 seconds
Mean completion of TT_2 = 0.9 seconds
Mean completion of TT_3 = 0.7 seconds

Have all sequences the same duration ? If not, explain why not.

Exercise 4: Total number of tasks of the RI

Use again the WPS as defined in Exercise 2. Estimate the total number of tasks which are executed in the following situation: The RI is about 30 minutes. Assume for the SUT execution times that they just fulfil the timeliness functions (i.e. the SUT is neither faster nor slower).

Exercise 5: Task lists for DEMO

Prerequisite: The DEMO system is installed according to Exercises 1 and 2 of Section 1.8.

Part 1
Two users are to be emulated. Write down the two task lists of Exercise 2 of the RI for the DEMO format. For each user DEMO needs 3 task lists: for the StP; the RI and the SR. Create the missing lists as follows for each user;

- for StP, copy the RI list
- for SR, copy the RI list .

Part 2
Perform the user emulation using the following three UNIX shell scripts as activity types 1 to 3.

File 'TT1s':

```
#!/bin/sh
#
sleep 1
# Simulates program run time
#
#== End of Procedure ==
```

File 'TT2s':

```
#!/bin/sh
#
sleep 2
# Simulates program run time
#
#== End of Procedure ==
```

File 'TT3s':

```
#!/bin/sh
#
sleep 3
# Simulates program run time
#
#== End of Procedure ==
```

These files are found in directory

CD/Sol-files/Mment.ch3/OSCPs-TTxs-shells .

Run the emulation by manually starting the DEMO module "demo". Print the DEMO logfile after running the DEMO module "demo2din".

Part 3

Investigate those parts of the logfile which belong to the RI and compute the following values:

- Mean value of the preparation times preceding TT_1, TT_2, TT_3
- Relative chain probabilities

Compare

- the computed relative chain frequencies with the values in the WPS and explain the reasons for the differences (if any)
- the computed mean preparation times values of the WPS and explain the reasons for the differences (if any).

Part 4

Investigate the logfile and estimate the times t_1 (begin of the RI) and t_2 (end of the RI).

Solutions

For solutions see file

CD/Solutions/Solutions-Section3-7.pdf .

4 Validation of the measurement results

The ISO method stipulates a systematic and comprehensive checking of each measurement run. This checking ensures that the measurement procedure was technically correct and that the results were reliable.

4.1 Validation of the computational results of the SUT

The first of the checks concerns the computational results of the SUT. For all tasks of the OP (the observation period, i.e. the time from t_1 to t_3) the task results produced by the SUT have to be compared to the predefined correct computational results which are part of the workload description (see Section 2.8).

This check includes intentionally those tasks which were submitted during the SR (the supplementary run, i.e. in the time from t_2 to t_3). This is to ensure that the SUT has performed real tasks in the SR and not only dummy tasks.

If all task results were complete and proved to be correct the measurement is acceptable with respect to the correctness of the SUT.

This assumes that a repeated measurement always starts with the same data configuration. However, an exception is when the influence of the data configuration is being tested, for instance if using a data base system with increasing amounts of data.

The validation of computational results cannot be done manually; a program is needed. If each task type always produces the same results such a program will be simple (file or string comparison). With input variation (see Section 2.7.1) such a program can be complex. If the output cannot be uniquely defined then the validation program may have to use heuristic algorithms. For example, the task might include a search in a digital online library, the content of which can be changed by other tasks of the workload.

If the SUT produces incorrect work then the measurement is invalid. Even if it were possible to compute performance values (in a later step of the ISO procedure) they should neither be rated nor published. The SUT has to be corrected and a new measurement carried out.

As recommended in Sections 3.4 and 3.5, the computational result file should be separated into two parts, one for the RI and one for the SR. Both these files will then be checked for a possible result "OK" or " NOT OK". This separation is for clarity in working.

For dynamic correctness checking without storing computational results see Section 4.4 .

4.2 Validation of the correctness of the working of the RTE

4.2.1 Three criteria

The second check concerns the correctness of the working of the RTE. It consists of the following three criteria:

- Relative chain frequencies
- Mean values of the preparation times (think times), and
- Their standard deviation values.

The actual values have to be computed from the measurement logfile and compared to the required values as defined in the WPS. If one or more of the measured values exceeds the defined relative limits (DELTA values, see Section 2.9.2) the measurement is invalid due to an inaccurate RTE. The measurement has to be repeated using a better working RTE.

4.2.2 The first criterion: Checking the relative chain frequencies

As recommended at the end of Section 3.4, the logfile should be separated into two parts, one for the RI and one for the SR. Both these files will then be checked for a possible result "OK" or " NOT OK". This separation is for clarity in working. Checking the RI part is now explained.

The actual relative chain frequencies have to be computed from the logfile of all chains started during the RI, i. e. the time from t_1 to t_2. Let $q_{meas}(i,l)$ be the computed relative frequency of chains of type l which were started during the RI by the group of user type i. $q(i,l)$ is the required relative frequency of chains of type l of user type i. This value is defined in the WPS. $DELTA_q(i,l)$ is the maximum allowed relative difference of the measured value $q_{meas}(i,l)$ to the set value $q(i,l)$. The actual relative difference is $DIFF_q(i,l)$. It is computed as follows:

$$DIFF_q(i,l) = (|\ q_{meas}(i,l) - q(i,l)|)/\ q(i,l) \qquad (4.1a)$$

Note: The vertical lines "|"..."|" mean the absolute value (i.e. ignoring the sign) of the value "..." . In the ISO/IEC 14756 this operator was unfortunately omitted in Annex B.4.1, due to a typing error. ■

$DIFFq(i,l)$ must not exceed $DELTA_q(i,l)$ for all user types and all chain types, i.e. for all combinations of i and l.

The ISO standard allows in the WPS the definition of DELTA values which are different for each combination of i and l. But in practice a unique $DELTA_q$ value is mostly used as cited in Section 2.9.2 . Therefore the inequality (4.1b) below uses $DELTA_q$ and not $DELTA_q(i,l)$.

$$DIFF_q(i,l) \leq DELTA_q \quad \text{for all user types } i = 1, 2, \ldots, n \quad \text{and for all chain types } l = 1, 2, \ldots, u \qquad (4.1b)$$

If this is true, the accuracy of the RTE is within the tolerances with respect to the chain generation within the RI. Performing this check means computing n*u values $DIFF_q(i,l)$ and comparing each of them to $DELTA_q$.

Example: In Fig. 2-8, Section 2.6, there are n=2 user types and u=3 chain types. 2*3=6 values have to be computed and compared to $DELTA_q$.

The SR part of the logfile has to be checked analogously to the RI part. For the relative chain frequencies, the functioning of the RTE is acceptable only if both the RI and SR measurements are acceptable. But there is a practical aspect. The SR is typically much shorter than the RI. Therefore, from statistical reasons, (4.1b) above is more likely to fail for the SR than for the RI. It is up to the person responsible for the measurement to decide whether or not the SR is acceptable for the relative chain frequencies.

4.2.3 The second criterion: Checking the preparation mean times

The same recommendation for separating the logfile as in Section 4.2.2 applies here. Checking the RI part for preparation times is now explained.

The actual preparation mean time values have to be computed from the logfile of all tasks of which the preparation times were started during the RI. Let $h_{meas}(i,j)$ be the computed mean preparation times of tasks of the j-th task type that were started during the RI by type i users. $h(i,j)$ is the required mean preparation time of type j tasks of type i users as defined in the WPS. $DELTA_h(i,j)$ is the maximum allowed relative difference of the measured value $h_{meas}(i,j)$ to the set value $h(i,j)$. The actual relative difference is $DIFF_h(i,j)$. It is computed as follows:

$$DIFF_h(i,j) = (|\ h_{meas}(i,j) - h(i,j)|)/\ h(i,j) \qquad (4.2a)$$

Note: The vertical lines "|"..."|" mean the absolute value. The omission of this operator, in Annex B.4.1 of ISO/IEC 14756 was unfortunately repeated in Annex B.4.2 . ■

$DIFF_h(i,j)$ must not exceed $DELTA_h(i,j)$ for all user types and all task types, i.e. for all combinations of i and j .

The ISO standard allows in the WPS the definition of DELTA values which are different for each combination of i and j. But in practice a unique $DELTA_h$ value is mostly used as cited in Section 2.9.2 . Therefore the inequality (4.2b) below uses $DELTA_h$ and not $DELTA_h(i,j)$.

$$DIFF_h(i,j) \leq DELTA_h \quad \text{for all user types } i = 1, 2,\ldots,n \quad \text{and for all task types } j = 1, 2,\ldots,m \qquad (4\text{-}2b)$$

If this is true the accuracy of the RTE is within the tolerances with respect to the mean preparation times of the tasks of the RI. Performing this check means computing n*m values $DIFF_h(i,j)$ and comparing each of them to $DELTA_h$.

Example: In Fig. 2-8, Section 2.6, there are n=2 user types and m=4 task types. 2*4=8 values have to be computed and compared to $DELTA_h$.

The SR part of the logfile has to be checked analogously to the RI part. For the mean preparation times, the functioning of the RTE is acceptable only if both the RI and SR measurements are acceptable. But there is a practical aspect. The SR is typically much shorter than the RI. Therefore, from statistical reasons, (4.2b) above is more likely to fail for the SR than for the RI. It is up to the person responsible for the measurement to decide whether or not the SR is acceptable for the preparation times.

4.2.4 The third criterion: Checking the standard deviations of the preparation times

The same recommendation for separating the logfile as in Section 4.2.2 applies here. Checking the RI part for standard deviation of the preparation time is now explained.

The actual standard deviations of the preparation times have to be computed from the logfile of all tasks of which the preparation times were started during the RI. Let $s_{meas}(i,j)$ be the computed standard deviation of the preparation time of type j tasks that were started during the RI by type i users. s(i,j) is the required standard deviation of the preparation times of the type j tasks of type i users as defined in the WPS. $DELTA_s(i,j)$ is the maximum allowed relative difference of the measured value $s_{meas}(i,j)$ to the set value s(i,j). The actual relative difference is $DIFF_s(i,j)$ is computed as follows:

$$DIFF_s(i,j) = (| s_{meas}(i,j) - s(i,j)|)/ s(i,j) \qquad (4.3a)$$

Note: The vertical lines "|"..."|" mean the absolute value. The omission of this operator, in Annex B.4.1 of ISO/IEC 14756 was unfortunately repeated in Annex B.4.3 . ■

$DIFF_s(i,j)$ must not exceed $DELTA_s(i,j)$ for all user types and all task types, i.e. for all combinations of i and j .

The ISO standard allows the definition of DELTA values in the WPS which are different for each combination of i and j. But in practice a unique value $DELTA_s$ value is mostly used as cited in Section 2.9.2 . Therefore the following inequality uses $DELTA_s$ and not $DELTA_s(i,j)$.

$$DIFF_s(i,j) \leq DELTA_s \quad \text{for all user types } i = 1, 2,\ldots,n \quad \text{and for all task types } j = 1, 2,\ldots,m \qquad (4\text{-}3b)$$

If this is true the accuracy of the RTE is within the tolerances with respect to the standard deviations of the preparation times of the tasks of the RI. Performing this check means to compute n*m values $DIFF_s(i,j)$ and to compare each of them to $DELTA_s$.

Example: In Fig. 2-8, Section 2.6, There are n=2 user types and m=4 chain types. 2*4=8 values have to be computed and compared to $DELTA_s$.

The SR part of the logfile has to be checked analogously to the RI part. For the standard deviations, the functioning of the RTE is acceptable only if both the RI and SR measurements are acceptable. But there is a practical aspect. The SR is typically much shorter than the RI. Therefore, from statistical reasons, (4.3b) above is more likely to fail for the SR than for the RI. It is up to the person responsible for the measurement to decide whether or not the SR is acceptable for the preparation times.

4.2.5 Remarks

While there is no doubt about how to calculate a mean value, various methods can be used for the standard deviation (so-called estimations). The method used in Annex B.5 of ISO/IEC 14756 is as follows.

Let $x_1, x_2, \ldots, x_N$ be the set of N samples. The ISO method uses the following formula for the estimation of the standard deviation $s_m(N)$ of this set of samples.

$$s_m(N) = \sqrt{(1/N) * \sum_{k=1}^{N} (x_k - m_m(N))^2} \qquad (4.4)$$

where

$$m_m(N) = (1/N) * \sum_{k=1}^{N} x_k \qquad (4.5)$$

is the mean value of the N samples .

4.3 Checking the statistical significance of the measurement results

4.3.1 Rationale for this check

Due to random preparation times and random choice of chain types, an ISO-type measurement is non deterministic. Additional influences are microbial differences in hardware operation, such as hard disc response times or network transmission times. Whenever a data processing system is driven by an ISO-type RTE timing of all events is random. Therefore the measured completion times vary by a certain amount. They are random variables. No measurement of the same SUT and the same ISO-type workload delivers exactly the same result. Each measurement result is an approximation of the

unknown true value. The question is: how good is this approximation ? An answer to this question may be found in the field of mathematical statistics, which offers statistical tests. There are many test methods. In the ISO standard a test suited to the actual problem is chosen.

4.3.2 The test

The ISO standard uses a so-called sequential test. To keep it simple this test is not applied to all details or all computed performance values. It is restricted to the mean execution time values. The philosophy behind this decision is that the most important performance values are computed from the execution times. And it is assumed that all performance values are significant if the execution time mean values (computed from the logfile) are statistically significant. The chosen test has to be performed separately for the execution times of each of the task types.

There are m task types. The test has to be performed for each task type either after the measurement or parallel to it. The test takes samples of the response times. An indicator "OK" or "NOT OK" shows for which of the task types the required statistical significance is achieved.

For those who have a good knowledge of mathematical statistics the test is easy to understand. "OK" indicates that the test was successful with respect for the task type under consideration. Successful means that the confidence interval "$T_{ME} \pm d$" covers the (unknown) true value. T_{ME} is the mean value computed from the samples. d is the half-width confidence interval. The test result ("covers" or "does not cover") is valid with a probability of 1−ALPHA where ALPHA is the confidence coefficient.

For those who are not familiar with mathematical statistics there follows an illustrative interpretation. The confidence interval can be interpreted as the accepted measurement error.

Example: In the WPS d = 1.0 sec was defined and T_{ME} = 4.9 sec has been computed from the logfile. "OK" coming from the test can be interpreted as the following message: "The (unknown) real mean value is within 4.9 ± 1.0 seconds". "NOT OK" means that the real mean value does not lie within the cited interval. The message coming from the test has some uncertainty. The probability of it being correct is 1−ALPHA . For instance, if in the WPS ALPHA = 0.05, the message "OK" means that with a probability of 95% T_{ME} is within the confidence interval $T_{ME} \pm d$. There is a 5% chance that it is not. See Section 2.9.3 for advice on setting ALPHA and d .

Please note that this explanation is only illustrative and not valid in a strict mathematical sense.

The detailed procedure of the test is described in Annex B.6 of ISO/IEC 14756. The test was developed and published by the University of Bonn (Germany) in about the 1980s. In 1991 it was included in "Part 1" of [DIN01]. In 1994 it appeared in the book [DIRLE02] in which also programs for performing the test are published.

Such programs are also contained in the ISO/IEC 14756, Annex E.6 and a program executing the test is also contained in the DEMO system (see the CD which is part of this book).

The test is described shortly as follows. Its steps have to be performed repeatedly. Let x_k be the k−th sample.

Step 1 (initial step): Set N to 2
Step 2: Compute the mean $m_m(N)$ of the first N samples
according to equation (4.5).
Step 3: Compute $s_m(N)$ of the first
N samples according to equation (4.4).
Step 4: Compute the variance

$$var(N) = (s_m(N))^2 \quad . \qquad (4.6)$$

Step 5: Check if

$$((U(ALPHA)/d)^2 * var(N)) > N \quad . \qquad (4.7)$$

If yes then increase N by 1 and go to Step 2;
if no then the test is finished having the result "OK".

If the test does not result with OK when all samples are used then this means "not OK". U(ALPHA) is the so-called $u_{1-ALPHA/2}$-quantil of the standard normal distribution N(0,1); for this quantity see ISO/IEC 14756 or statistic handbooks. ALPHA is the confidence coefficient as defined in the WPS. d is the half-width confidence interval as defined in the WPS.

4.3.3 Application of the sequential test

The test has to be applied to the tasks started within the RI. If the logfile has been separated into 2 parts, as recommended in Sections 3.4 and 3.5 then the sequential test has to be applied to the RI part of the logfile.

The application of the test is easy. For task type j proceed as follows.

Input to the test are the values of ALPHA and of d(j) which are to be taken from the workload. Additional input is the set of statistically independent execution time samples of task type j . For simplicity all measured execution time values of the RI of task type j in their chronological order are taken as samples. Rationale: In practice, due to the complexity of the events in the SUT, you can assume a sufficiently statistical independence of the execution times in spite of taking them in chronological order. In a mathematical sense there is no statistic independence. But experience shows that this approximation is usually acceptable in the field of ISO-type performance measurements.

For checking the overall significance the test has to be executed by a program for each of the m task types. If all m tests deliver "OK" the statistical significance of the measurement is sufficient. If at least one task type has "NOT OK" the measurement is not acceptable, and it has to be repeated.

There are many reasons for a "NOT OK". A typical reason could be "too few samples". I.e. the RI was too short. A new measurement having a longer RI has to be performed. Another reason could be that the SUT was severely overloaded with execution times fluctuating strongly and perhaps periodically. This could be a situation in which significant execution time mean values and consequently performance values do not exist.

4.3.4 Fast computation of mean value and variance

The mean value $m_m(N)$ and variance $var(N)$ have to be computed in each run through the loop of the test. This needs the more time the larger N is. To solve this problem the author developed a method for faster computation and published it in [DIRLE02]. The method uses recursive formulae for computing mean and variance as follows.

Let $x_1, x_2, \ldots, x_{N+1}$ be the set of $N+1$ samples and $m_m(N)$ the mean value of the first N samples. The mean value $m_m(N+1)$ can be computed by the formula

$$m_m(N+1) = (N * m_m(N) + x_{N+1}) / (N+1) \quad . \qquad (4.8a)$$

The start condition for this recursive formula is

$$m_m(1) = x_1 \quad . \qquad (4.8b)$$

The variance of $N+1$ samples can, using equations (4.4) and (4.6) be written as

$$var(N+1) = qs(N+1) / (N+1) \quad , \qquad (4.9)$$

where $qs(N+1)$ is

$$qs(N+1) = \sum_{k=1}^{N+1} (x_k - m_m(N+1))^2 \quad . \qquad (4.10)$$

This term can also be computed by the following formula:

$$qs(N+1) = qs(N) + (x_{N+1})^2 - (N+1) * (m_m(N+1))^2 + N * (m_m(N))^2 \qquad (4.11a)$$

The start condition for this recursive formula is

$$qs(1) = 0 \quad . \qquad (4.11b)$$

Using the formula (4.8a) for computing mean values and (4.11a) and (4.9) for variance is much faster than using the generic formulae (4.5) and (4.4). This method is implemented in DEMO.

4.4 Summary of the validation procedure

The validation procedure has three parts.

Part 1: Check the correct working of the SUT
(see Section 4.1)
Part 2: Check the correct working of the RTE.
There are three criteria.
- Criterion 1 checks the `n*u` relative chain frequencies (see Section 4.2.2).
- Criterion 2 checks the `n*m` preparation mean times (see Section 4.2.3).
- Criterion 3 checks the `n*m` preparation time standard deviations (see Section 4.2.4).

Part 3: Check of the statistical significance of the `m` mean execution times
(see Section 4.3.3).

Parts 2 and 3 can only be performed when the measurement run is completed (i.e. only after t_3).

Checking according to Part 1 can either be performed after the end of the measurement (t_3) or it can be performed "dynamically" during the measurement run.

Such a dynamic check works as follows. Each computational result is checked as soon as it is delivered by the SUT. The check has to be performed by the RTE. It cannot be performed by the SUT itself because this would produce an additional component of the workload. Implementing a dynamic check typically increases the RTE CPU loading but avoids storing the computational results. It is recommended to not use dynamic check without storing computational results though ISO/IEC 14756 allows it.

Part 1 is usually performed after the measurement run. It can then be performed either by the RTE or by the SUT itself or by another computer. The execution time of Part 1 is of little importance, as long as it is not too excessive.

A diagram of the checking course is shown in Fig. 4-1 .

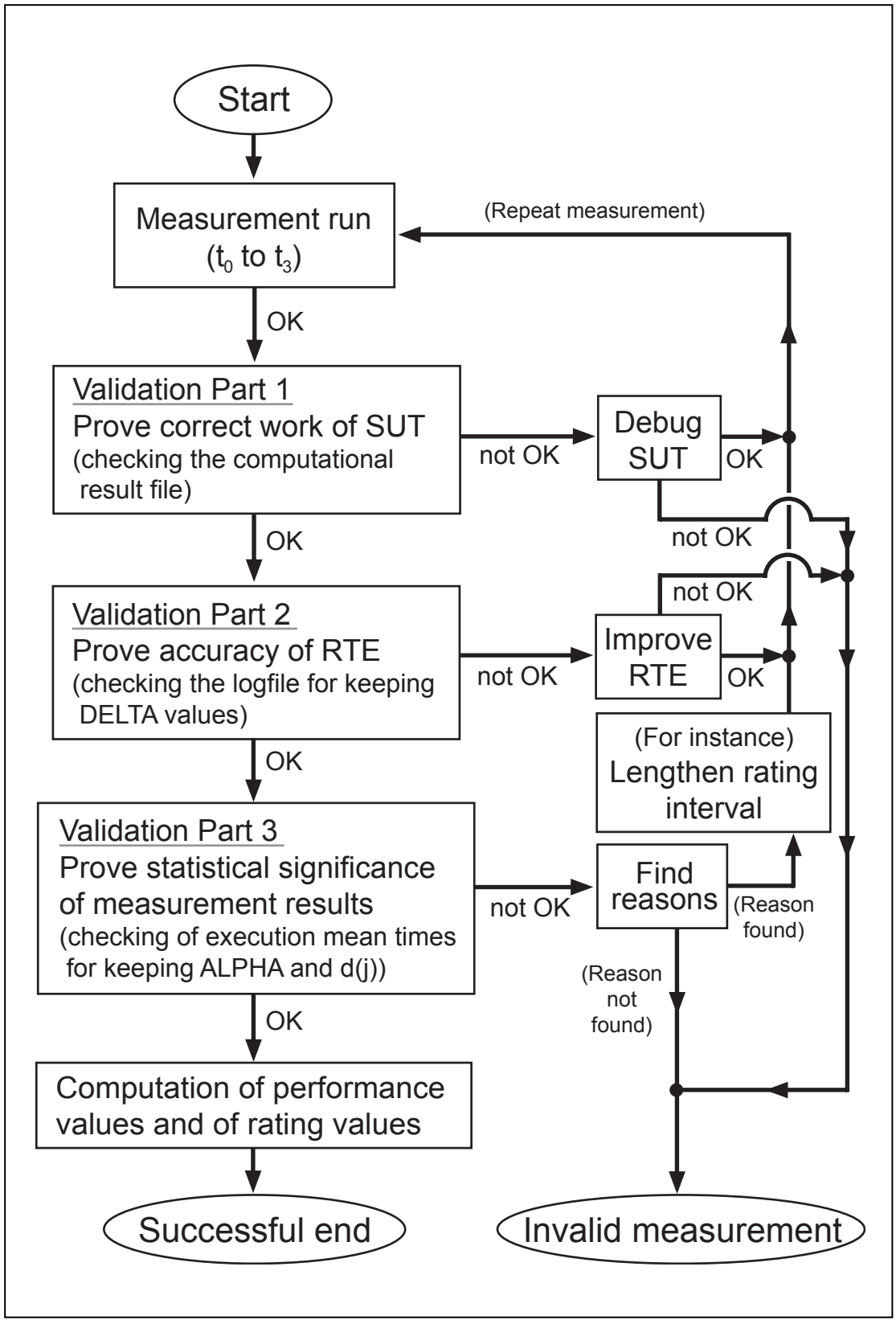


Fig. 4-1 The ISO-type validation of a measurement

4.5 Exercises

Preparation:
Use the WPS of Chapter 3, Section 3.7, Exercise 2 and the same task lists. But change the activity types to `TT1c, TT2c, TT3c` . These UNIX shell scripts are as follows:

`Shell TT1c:`

```
#!/bin/sh
#
# Shell procedure TT1c
mkdir $$
cd $$
# Creating a directory having a unique name,
# which is the actual UNIX process number.
echo TT1.$$ > TT1-Res
cat TT1-Res ../infile1 >> ../TT1.out
# Bringing the name of the called shell
# and the actual UNIX process number - as
# an identifier of the run - to the output file.
sleep 1
# Simulates program runtime
rm -f *
cd ..
rmdir $$
# Cleaning temporary files and the directory.
#
# == End of procedure ==
```

Explanation: This shell produces two lines and adds them to the output file `"TT1.out"`. The first line contains the text `"TT1.xxxx"` where `xxxx` is the UNIX process number. This number is a unique identifier of the task executed. The second line contains the text stored in the input file `"infile1"`. (This text is `"Printing infile1"`). The shells `TT2c` and `TT3c` work analogously.

Note (concerning DEMO): The names of shells, i. e. the input strings of activity types can be chosen arbitrarily, but due to the actual file formats used by the module `"demo2din"` they are limited to a maximum of 6 characters. For using longer strings, `demo2din` and its co-operating DEMO modules would have to be rewritten. ■

`Shell TT2c:`

```
#!/bin/sh
#
# Shell procedure TT2c
mkdir $$
cd $$
# Creating a directory having a
# unique name,
# which is the actual UNIX process
# number.
echo TT2.$$ > TT2-Res
cat TT2-Res ../infile2 >> ../TT2.out
# Bringing the name of the called
# shell
# and the actual UNIX process number
# - as
# an identifier of the run - to the
# output file.
sleep 2
# Simulates program runtime
rm -f *
cd ..
rmdir $$
# Cleaning temporary files and the
# directory.
#
# == End of procedure ==
```

`Shell TT3c:`

```
#!/bin/sh
#
# Shell procedure TT3c
mkdir $$
cd $$
# Creating a directory having a unique
# name,
# which is the actual UNIX process number.
echo TT3.$$ > TT3-Res
cat TT3-Res ../infile3 >> ../TT3.out
# Bringing the name of the called shell
# and the actual UNIX process number - as
# an identifier of the run - to the output
# file.
sleep 3
# Simulates program runtime
rm -f *
cd ..
rmdir $$
# Cleaning temporary files and the
# directory.
#
# == End of procedure ==
```

These shells are found in the directory
CD/Sol-files/Mment-ch4/OSCPs-TTxc-shells/ .
The input files are found in the directory
CD/Sol-files/Mment-ch4/input-files-TTxc-Mment/ .

Perform the user emulation by manually starting the DEMO module "demo". Create the logfile by manually starting the module "demo2din".

Exercise 1: Check the correct working of the RTE

Part 1
Analyse the logfile. For its file format see file
CD/DEMO-20/DEMO-manual/file-formats.txt .
Check the RI (i.e. the interval t_1 to t_2) to see whether the $DELTA_q$ criteria are fulfilled.

Part 2
Check the RI to see whether the $DELTA_h$ and $DELTA_s$ criteria are fulfilled. You can perform the computations manually or you can start the DEMO module "STK.CON".

Exercise 2: Check the correct working of the SUT

Check whether all submitted tasks had been executed. Check the correctness of the computational results.

Exercise 3: Check the statistical significance

Set ALPHA = 0.20 and d_{rel} = 0.25 . Check, by manually starting the DEMO module "DINREA", whether there is statistical significance.

Exercise 4: Demonstration of the incorrect working of the SUT

Repeat the measurement but apply the following modification: Before starting the measurement login to the SUT as an additional "user2". Insert intentionally the following error: While the measurement runs in the RI delete by using the UNIX command "rm" the file "infile2" in the home of "user2". After finishing the measurement check manually if the stored computational results of all tasks of the RI and the SR exist and are correct.

Exercise 5: Statistical significance when using more rigorous values of the criteria

Use the logfile of the measurement of Exercise 1 that consists of the files DIN.DAT and ZEIT. Reduce stepwise ALPHA and d_{rel}. Check for each step - by starting manually the DEMO module "DINREA" - whether there is statistical significance.

Solutions
For solutions see file
CD/Solutions/Solutions-Section4-5.pdf .

5 Computing the ISO performance values from the measurement result file (logfile)

5.1 Overview of the ISO performance terms

Performance P is a triple of three terms

$$P = (B, T_{ME}, E) \qquad (5.1)$$

where

- B is the (total) throughput vector
- T_{ME} is the mean execution time vector
- E is the timely throughput vector

It is important to understand that performance is the set of these three terms. The reason is that no term can be computed from the others. In the general case of the SUT being a black box the three terms are independent of each other.

These terms are determined for each task type. I.e. each is a tuple of as much values as there are task types. For instance, for m = 4 task types the performance is described by the three terms each subsuming four subterms, i.e. by 12 subterms. In general the following holds:

$$B = (B(1), B(2), ..., B(m)) \qquad (5.2)$$
$$T_{ME} = (T_{ME}(1), T_{ME}(2), ..., T_{ME}(m)) \qquad (5.3)$$
$$E = (E(1), E(2), ..., E(m)) \qquad (5.4)$$

P consists of 3*m subterms. This is a complex but very powerful definition of performance.

The values of P are computed from the logfile records, concerning the rating interval (RI). If following the recommendation of Sections 3.4 and 3.5 the logfile is separated into two parts. For the computation of P is only the RI part of the logfile is needed.

5.2 The "total throughput vector" B

B is the set of m terms B(j) where j is the current number of the task type. Every B(j) refers to the duration T_R of the RI

$$T_R = t_2 - t_1 \qquad . \qquad (5.5)$$

B(j) is the number of type j tasks submitted to the SUT by the RTE per time unit.

$$B(j) = b(j) / T_R \qquad (5.6)$$

In the above formula $b(j)$ is the total number of tasks of type j submitted by all emulated users to the SUT during the RI. The set of the m terms $B(j)$ yields the throughput vector as defined in equation (5.2) above.

The computation of B is simple. Analyse the RI part of the logfile. Counting the number of tasks of each task type yields the m values $b(j)$. According to equation (5.6) above each of these values has to be divided by T_R . This yields the m values $B(j)$. This set of values is the throughput vector B as defined in equation (5.2) .

5.3 The "mean execution time vector" T_{ME}

T_{ME} is the set of m terms $T_{ME}(j)$ where j is the current number of the task type. Every $T_{ME}(j)$ refers to the RI and $T_{ME}(j)$ is the mean execution time of all type j tasks submitted to the SUT by all emulated users within the RI.

$T_{ME}(j)$ is simple to compute. Analyse the RI part of the logfile. The execution times of all type j tasks within the RI. These times are $t_{ET}(j,1)$, $t_{ET}(j,2)$,..... . The total number of those values is $b(j)$. Therefore the last element in the series will have the symbol $t_{ET}(j,b(j))$. Adding them and dividing by $b(j)$ yields the mean value $T_{ME}(j)$.

$$T_{ME}(j) = ((t_{ET}(j,1) + t_{ET}(j,2) + ...+t_{ET}(j,b(j))) / b(j) \qquad (5.7)$$

Performing this simple procedure for all m task types yields the "mean execution time vector T_{ME} in equation (5.3) above.

5.4 The "timely throughput vector" E

5.4.1 The principle of E

Although, E is a somewhat unusual term, it is easy to understand.

As explained in Section 5.2 $b(j)$ is the number of type j tasks submitted to the SUT during the RI. With regard to the according timeliness function (see Section 2.5) some tasks may not have been executed in time. Let $e(j)$ be the total number of timely executed tasks of type j tasks during the RI. The maximum value of $e(j)$ may be achieved if all type j tasks have been executed in time. Therefore,

$$e(j) \leq b(j) \quad . \qquad (5.8)$$

The "timely throughput" E(j) of type j tasks is the number per time unit. Therefore,

$$E(j) = e(j) / T_R \quad . \qquad (5.9)$$

From this definition and equation (5.8) above follows that

$$E(j) \leq B(j) \qquad (5.10)$$

This inequality is the mathematical representation of the fact that the rate of timely executed tasks cannot exceed the total rate of tasks (with respect to j type tasks).

The procedure for computing e(j) (for getting E(j) by use of equation (5.9) above) is somewhat sophisticated. It is explained in the Section 5.4.2 .

5.4.2 Computing e(j)

The procedure for computing e(j) was developed in the 1980s by the performance subgroup of the German National Standardisation Committee DIN. It was adopted for the German National Standard DIN 66273 [DIN01] in the early 1990s. It appeared also in [DIRLE02] in 1994.

This procedure is fully explained in ISO/IEC 14756, Annex B.3. You are advised to read this annex. A short introduction is given below as an example.

We consider and define a task type. Its timely function TF (see Section 2.5) has z = 2 time classes with the following values:

TF:

z = 2		
k	$g_T(k)$	$r_T(k)$
1	5.00 sec	0.80
2	10.00 sec	1.00

Fig. 5-1 Example values of a timeliness function having two time classes

Its time classes are:

class 1: 0.00 to 5.00 seconds
class 2: 5.00 to 10.00 seconds

The total number of tasks (of the considered task type) submitted to the SUT during the RI is assumed to be b = 5. The timeliness function implies that in the first time class there should be

$$b * r_T(1) = 5 * 0.80 = 4 \text{ tasks} ,$$

i.e. 80% of the tasks. In the second time class there should be

$$b * (r_T(2) - r_T(1)) = 5 * (1.00 - 0.80) = 1 \text{ task} ,$$

i.e. 20 % of the tasks. These numbers are the reference values:

$$b_{RefClass}(1) = 4$$
$$b_{RefClass}(2) = 1$$

For computing the number of tasks counted as "in time" we sort the execution time values and put them into the time classes. Then we compare the number in each class with its reference value.

Does the actual number of execution times in a class not exceed the reference value, then the actual number is to be counted for this class. If this number is greater than the reference value then the reference value is to be counted for this class; additionally any surplus tasks are put into the next class as a bonus of timely tasks.

This counting has to be performed for each class, beginning with the first class, then for the second class and so on. The total number of timely executed tasks is the sum of the counts for each class. Four cases are shown below as examples.

Case 1

```
Measured times (sec) : 2.0, 9.5, 1.0, 3.0, 10.1
Sorted into class 1  : 3 tasks (values 2.0, 1.0, 3.0)
Sorted into class 2  : 1 task  (value 9.5)
Counted for class 1  : 3 tasks (less than the reference value 4)
Counted for class 2  : 1 task  (equals the reference value 1)
Timely executed      : e = 3 + 1 = 4
```

Case 2

```
Measured times (sec) : 2.0, 4.5, 1.0, 10.3, 3.0
Sorted into class 1  : 4 tasks (values 2.0, 4.5, 1.0, 3.0)
Sorted into class 2  : none
Counted for class 1  : 4 tasks (equals the reference value 4)
Counted for class 2  : none    (less than the reference value 1)
Timely executed      : e = 4 + 0 = 4
```

Case 3

```
Measured times (sec) : 2.0, 8.5, 1.0, 3.0, 7.0
Sorted into class 1  : 3 tasks (values 2.0, 1.0, 3.0)
Sorted into class 2  : 2 tasks (values 8.5, 7.0)
Counted for class 1  : 3 tasks (less than the reference value 4)
Counted for class 2  : 1 task  (1 is the reference; if more than
                                one are in the class only one can
                                be counted. There is no upper
                                class which could use the
                                overflowing one; it is lost.)
Timely executed      : e = 3 + 1 = 4
```

Case 4

```
Measured times (sec) : 2.0, 4.5, 1.0, 3.0, 4.1
Sorted into class 1  : 5 tasks (all values)
Sorted into class 2  : none    (no value)
Counted for class 1  : 4 tasks (4 is the reference; if more than 4
                                are in the class only 4 can be
                                counted; the excess value goes to
                                the next class as a bonus)
Counted for class 2  : 1 task  (1, this is the bonus from class 1)
Timely executed      : e = 4 + 1 = 5
```

It is important to understand that this algorithm only counts the number of timely executed tasks. It does not decide individually whether a task is executed timely, except for those beyond the uppermost time class. They are clearly tasks which are not timely.

The algorithm is implemented in DEMO. For subroutines see [DIN01], [DIRLE02] and [ISO14756].

5.4.3 The "timely throughput vector" E

The number `e(j)` of timely executed type j tasks is determined according to Section 5.4.2. The timely throughput `E(j)` of type `j` tasks is to be computed according to equation (5.9) above. Performing this for all `m` task types yields the "timely throughput vector" `E` as defined in equation (5.4) above.

5.5 Exercises

Exercise 1: Computing P from the logfile

Compute P (performance) manually from the logfile below. It consists of the two files `ZEIT` and `DIN.DAT`. It was produced by the measurement of Exercise 1, Section 4.5 . These files are found in directory

`CD/Sol-files/Mment-ch4/ARCHIVE-TTxs-Mment/` .

For the sake of simplicity overlook that this logfile was produced by applying so-called individual rating intervals and use for duration T_R of the RI the mean value of the individual rating intervals of the 2 users.

Apply the following timeliness function TF_1 for all task types:

TF_1 :

z = 2

k	$g_T(k)$	$r_T(k)$
1	1.5 sec	0.80
2	2.5 sec	1.00

file ZEIT

```
                1   1          116.427 t_1
                2   1          112.521 t_1
   *
                1   1          233.050 t_2
                2   1          225.125 t_2
   *
```

file DIN.DAT

```
    1       1   1   1   1   116.449   124.467   125.509 0       X
    2       1   1   1   2   125.513   135.544   137.573 0       X
    3       1   1   2   2   137.576   154.599   156.618 0       X
    4       1   1   2   3   156.622   166.637   169.668 0       X
    5       1   1   1   1   169.672   188.686   189.745 0       X
    6       1   1   1   2   189.748   195.780   197.799 0       X
    7       1   1   1   1   197.803   202.859   203.884 0       X
    8       1   1   1   2   203.888   211.903   213.925 0       X
    9       1   1   1   1   213.928   214.943   215.990 0       X
   10       1   1   1   2   215.994   231.025   233.048 0       X
   11       2   1   2   2   112.544   112.568   114.616 0       X
   12       2   1   2   3   114.619   121.646   124.675 0       X
   13       2   1   1   1   124.679   140.695   141.721 0       X
   14       2   1   1   2   141.724   156.738   158.778 0       X
   15       2   1   2   2   158.782   165.841   167.873 0       X
   16       2   1   2   3   167.877   174.905   177.933 0       X
   17       2   1   2   2   177.937   179.951   181.973 0       X
   18       2   1   2   3   181.977   193.987   197.029 0       X
   19       2   1   1   1   197.032   211.060   212.082 0       X
   20       2   1   1   2   212.086   223.102   225.123 0       X
*
```

Note: It is expected that your measurement produces results that differ a little from those above because of differences due to probabilistic events in detail. ■

Exercise 2: Similar workload (slower users)

Repeat the measurement of Exercise 1, Section 4.5 but double all preparation times in the task lists. Compute P manually from the logfile. For the sake of simplicity overlook - as in Exercise 1 - that the logfile is produced by applying so-called individual rating intervals. For duration T_R of the RI use the mean value of the rating intervals of the 2 users. Apply the timeliness function TF_1 above for all task types. Compute P and explain all changes.

Exercise 3: Similar workload (faster users)

Repeat the measurement of Exercise 1, Section 4.5 but halve all preparation times in the task lists. Compute P manually from the logfile. For the sake of simplicity overlook - as in Exercise 1 - that the logfile is produced by applying so-called individual rating intervals. For duration T_R of the RI use the mean value of the rating intervals of the 2 users. Apply the timeliness function TF_1 above for all task types. Compute P and explain all changes.

Solutions

For solutions see file

CD/Solutions/Solutions-Section5-5.pdf .

6 The Urn Method

6.1 General

As explained in Section 3.3, the measurement consists of three phases:

- StP (stabilisation phase)
- RI (rating interval)
- SR (supplementary run).

At the end of StP (time t_1) a user is mostly not at the start of a new chain. He is in a random position within his chain. At the end of the RI (time t_2) a similar situation occurs. At this time the users are not at the ends of their chains. In consequence the RI contains typically for all users incomplete chains. This makes it difficult to satisfy the RTE correctness criteria $DELTA_q$, $DELTA_h$ and $DELTA_s$. To satisfy these criteria a very long RI would be necessary. Shortening the RI is possible by use of the so-called individual rating intervals (see Section 6.2 below). This method has the following advantage: the RI includes only complete chains.

The use of individual rating intervals is necessary for the application of the so-called urns (see Sections 6.3 ff.). The use of such urns solves the problem of finite random sequences which was discussed in Section 3.6.3. It was pointed out that in general the relative frequencies of a finite random chain sequence differ from the required values in the WPS. Also the mean values and standard deviations of a finite number of preparation times differ from the values required. The shorter the RI and random sequence, the greater are the differences. The concept of "urns" solves this problem. It yields the same relative chain frequencies as in the WPS, and also for the preparation time mean values and their standard deviations.

The concept of "urns" including the concept of "individual rating intervals" is the so-called Urn Method, the basics of which were developed by the author who published it in [DIRLE01]. It appeared again in [DIRLE02]. The Urn Method is explained in the following sections.

6.2 Introduction to the concept of individual rating intervals

6.2.1 Defining the individual rating intervals

The ISO measurement procedure as explained in the previous chapters uses a common RI for all users, lasting from t_1 to t_2 . Contrary to this now for each user a new, individual RI has to be defined as follows.

The time t_1 is the planed end of the StP and the beginning of the RI. If at this moment the actual task of a particular user has not been fully processed, then the start of this new individual RI (time t_{1ind}) has to be delayed. This t_{1ind} can only be set when the current task is finished. However this time is inappropriate if the task just processed was not the last task in its chain. Because further tasks in this chain have still to be processed, t_{1ind} needs to be delayed still further until it coincides with the end of the last task in the chain. Thus the new individual RI starts with processing of a new chain, i.e. with the completion

of the last task in the current chain. If the last task of the current chain has the task mode NOWAIT then the individual RI starts when this task is submitted.

An analogous situation happens at t_2. The time t_2 is the planed end of the RI. For each user the end of the new individual RI (time t_{2ind}) has to be delayed. This t_{2ind} has to be set when the processing of the last task of the current chain has finished, i.e. with the completion of its last task. If the last task of the current chain has the task mode NOWAIT then the individual RI ends when this task is submitted.

In this way an individual RI contains only complete chains.

Using such an individual RI for each of the emulated users is necessary for achieving absolute correctness with respect to the $DELTA_q$ criterion. Additionally it ensures that no incomplete chains arise for the computation of the performance values.

The duration of the individual rating interval of a particular user is

$$T_{Rind} = t_{2ind} - t_{1ind} \qquad (6.1)$$

6.2.2 Modifying the computation of performance values

As a consequence of using individual RIs, the computation of performance vectors B, T_{ME} and E, presented in Chapter 5, have to be modified.

6.2.2.1 Computation of B (total throughput)

As t_1 and t_2 have been superseded by t_{1ind} and t_{2ind}, the number of tasks of each task type have as a first step to be counted separately for each user from its individual RI. These numbers are named $b_{ind}(j)$ where j is the task type number.

The total number of users is N_{tot} (see equation (2.4) in Section 2.6). Consequently there are

N_{tot} values $b_{ind}(1)$, N_{tot} values $b_{ind}(2)$,...., N_{tot} values $b_{ind}(m)$.

Analogously to equation (5.6) in Section 5.2, we compute the individual throughput values $B_{ind}(j)$, where j is the number of the task type and T_{Rind} is the individual RI of a particular user:

$$B_{ind}(j) = b_{ind}(j) \;/\; T_{Rind} \qquad (6.2)$$

This yields

N_{tot} values $B_{ind}(1)$, N_{tot} values $B_{ind}(2)$, ... , N_{tot} values $B_{ind}(m)$.

Then as a second step

the $B_{ind}(1)$ values have to be totalled for all users, yielding $B(1)$ and
the $B_{ind}(2)$ values have to be totalled for all users, yielding $B(2)$ and
etc. ..... and
the $B_{ind}(m)$ values have to be totalled for all users, yielding $B(m)$.

This is the B vector.

6.2.2.2 Computation of T_{ME} (mean execution time)

The tasks for the computation of the mean execution times have to be taken from the set of the N_{tot} individual RIs. Then the computation is the same as shown in Section 5.3. This is the mean execution time vector T_{ME}.

Do not try to compute T_{ME} analogously as to B (as above), because doing so would compute $T_{ME}(j)$ as a mean of means. But the mean of means is not the mean of the entirety.

6.2.2.3 Computation of E (timely throughput)

As t_1 and t_2 have been superseded by t_{1ind} and t_{2ind}, the tasks for the computation of the timely throughput have as a first step to be taken separately for each user from its individual RI. Then for each user the m numbers $e_{ind}(j)$ of timely tasks (where j is the number of the task type) have to be estimated according to Section 5.4.2. The total number of users is N_{tot}. Consequently there are

N_{tot} values $e_{ind}(1)$, N_{tot} values $e_{ind}(2)$ and so on and N_{tot} values $e_{ind}(m)$.

Analogous to equation (5.9) in Section 5.4.1, we compute the individual timely throughput values $E_{ind}(j)$ where j is the number of the task type and T_{Rind} is the individual rating interval of a particular user:

$$E_{ind}(j) = e_{ind}(j) / T_{Rind} \qquad (6.3)$$

This yields

N_{tot} values $E_{ind}(1)$, N_{tot} values $E_{ind}(2)$, ... , N_{tot} values $E_{ind}(m)$.

Then as a second step

the $E_{ind}(1)$ values have to be totalled for all users, yielding $E(1)$ and
the $E_{ind}(2)$ values have to be totalled for all users, yielding $E(2)$ and
etc. ..... and
the $E_{ind}(m)$ values have to be totalled for all users, yielding $E(m)$.

This is the E vector.

This is an illustrative description of the modified computation of the performance values B, T_{ME} and E. For a more formal and detailed description see Section 6.5, which should be used when programming an RTE with individual RIs.

6.2.3 Modifying the definition of the end of the SR

Using individual RIs implies a modified definition of t_3 , the end of the SR. Analogous to t_2 we define an individual SR for each user the end of which is t_{3ind} , i.e. the moment when all tasks which were submitted before t_{3ind} are finished. The duration T_{Sind} of the individual SR of a particular user is

$$T_{Sind} = t_{3ind} - t_{2ind} \qquad (6.4)$$

When all users have reached the end of their individual SRs this common time is t_4 . The recording of the logfile can be terminated and the RTE can be shut down, the logfile having been saved.

6.2.4 Overlapping of individual RIs

The experience of many measurements using individual RIs is that they may overlap, but not too much. ISO/IEC 14756, Clause 11.2, limits the amount of overlapping of the IRs as follows:

- The time difference of the earliest and the latest t_{1ind} shall not exceed 10% of T_{MR} .
- The maximum T_{Rind} may not exceed 120% of T_{MR} .
- The maximum T_{Sind} may not exceed 150% of the minimum T_{Sind} .

To check whether the overlapping satisfies ISO/IEC, Clause 11.2, the mean RI duration T_{MR} has to be computed from the individual RI durations T_{Rind} of all users.

6.3 Explaining the concept of "urns"

6.3.1 Toleration of the Urn Method by ISO/IEC 14756

Using individual RIs is the first step for solving the problems discussed in Section 6.1 . But the following problem remains: the generation of chain sequences and the execution time values by using a conventional random generator delivers pseudo-random sequences of chains and pseudo-random values of preparation times. Their relative frequencies and mean values differ from the predefined values in the WPS. This is unavoidable because the sequences are of finite length, as already mentioned in Section 6.1 . Only if the RI is very long can the measured values approximate the predefined values in the WPS.

The urn concept solves the problem of the differences. It eliminates the differences with respect to the RTE correctness criteria $DELTA_q$, $DELTA_h$ and $DELTA_s$ (see Section 4.2). The urn concept, not being a part of ISO/IEC 14756 is not described there. But it does not

conflict with the standard. Therefore it can be used for an ISO-type measurement. The only prerequisite for the urn concept is the use of individual RIs. This concept is tolerated in the ISO standard.

The Urn Method, being the combination of the individual RI concept and the urn concept is also tolerated in the ISO standard.

6.3.2 The urns

The Urn Method uses pregenerated task lists (see Section 3.2). Although it would be possible to implement the urn method for dynamic task generation, a typical application uses pregenerated task lists. First we have to generate chain sequences, then task lists and then preparation time values.

6.3.2.1 Generating chain sequences

The following example shows how the Urn Method generates a chain sequence. For `q` values we choose the chain probabilities of user type 1 (see Fig. 2-14 in Sect. 2.7.5). The values are

```
q(1,1) = 0.10
q(1,2) = 0.50
q(1,3) = 0.40    .
```

From these values it is easy to see that the minimum number of chains which realises these relative frequencies is 10. We take an urn and 10 balls. On one bowl is written "chain type 1". On five bowls is written "chain type 2". On four bowls is written "chain type 3". We put these balls into the urn and shake it. Then we invite a good fairy to take at random one ball out of the urn, the another, and so on. When the urn is empty the fairy has generated a random sequence of 10 chain type numbers. The relative frequencies of chain types equal exactly the `q` values above. Such a sequence is called "cycle". To get a longer sequence of chain types we put the 10 bowls back into the urn and generate another sequence of 10 numbers to add to the first. So now we have generated a random sequence of 20 chain type numbers. Alternately we can put two identical sets of 10 bowls into the urn and ask the fairy to take all 20 out randomly. We see that the length of the RI can be 10 or 20 or 30 or any multiple of 10 chains, that is, the length can be any number of cycles.

In the light of mathematical statistics, the randomness of these sequences is somewhat reduced. The reason is that the relative frequencies of the bowls change when a bowl is removed from the urn. But practice has shown that this small reduction of randomness has no negative influence on the measured performance values.

6.3.2.2 Generating preparation time sequences

We can also use this concept of the urn for generation of random think time sequences, as explained by the following example. We use the chain definitions of Fig. 2-12 in Section 2.7.4. In the 10 chains of a cycle there are 17 tasks as follows (see the example shown in Fig. 3-1 in Section 3.1):

9 tasks of task type 1
5 tasks of task type 2
2 tasks of task type 3
1 task of task type 4

According to the 4 task types we need 4 urns.

Into the first urn we put 9 bowls. On each of them has been written a preparation time value. We choose these 9 values so that their mean value equals the h-value in the WPS. Additionally we choose them so that their standard deviation equals the s-value in the WPS. We shake the this urn and ask the fairy again to do her work. When the urn is empty a random sequence of 9 think times is generated. The mean value and the standard deviation of this sequence exactly equals the required value in the WPS.

Similarly, we can also generate the think times of the tasks of types 2 to 4 using the next three urns. When the fairy comes to the 4th urn, she can have a rest, as the urn contains only one ball and we know the preparation time of this task.

To realise the urn method we need for each user one chain urn and m think time urns. For a large number of users the total number of urns can be large. But this causes no problems. Each urn in the RTE task list generator program is realised by a data structure such as a little array. The total amount of storage needed is no problem for a present-day computer.

6.3.3 Generation of a set of preparation time values for filling a preparation time urn

For generating the preparation time sequences (as explained in Section 6.3.2.2) the following problem has to be solved: Y non-negative numbers

$$x_1,\ x_2, \ldots\ldots, x_Y$$

have to be found. The mean value of this set has to equal the defined value h and the standard deviation has to be equal the defined value s.

As ISO/IEC does not specify preparation time distributions, one is free to solve the problem as one likes. A solution, involving a unique distribution type, was developed by the author in the early 1990s and published in [DIRLE02]. The solution can be applied here.

Compute the value Δ using the formula (6.5):

$$\Delta = (s * \sqrt{3}) / (\sqrt{Y^2 - 1}) \qquad (6.5)$$

If Y is an odd number, the set of the Y numbers is

$$h-(Y-1)*\Delta\ ,\ldots\ ,h-(4*\Delta),\ h-(2*\Delta),\ h,\ h+(2*\Delta)\ ,h+(4*\Delta)\ ,\ldots\ ,h+(Y-1)*\Delta\ .$$

(6.6a)

If Y is an even number, the set of the Y numbers is

$$h-(Y-1)*\Delta,\ldots,\ h-(3*\Delta),\ h-\Delta,\ h+\Delta,\ h+(3*\Delta),\ldots,\ h+(Y-1)*\Delta \quad . \qquad (6.6b)$$

The proof is given in [DIRLE02]. The formula to be used for the standard deviation is given in (4.4) in Section 4.2.5 .

One must note that a unique distribution having a positive mean h can contain negative values in case of the standard deviation s is too large. Therefore defining too big s values in the WPS yields negative preparation times. This has to be checked before filling the urns. If there are negative preparation times the task list cannot be executed. The s values have to be reduced and new task lists generated. This difficulty can be avoided as follows:
The leftmost terms of (6.6a) and (6.6b) must not be negative. Solving these inequations yields

- if there are Y=2 values, then $s \leq h$,
- if the number Y of values is very large, then $s \leq 0.57 * h$.

Therefore as a rule of thumb for the Urn Method, do not set s greater than 50% of the mean value h.

6.4 Experiences from applying the Urn Method

The particular advantage of the Urn Method is that the DELTA criteria are fulfilled and also that the RIs are as short as possible. The urn method is easy to use when applying pregenerated task lists (see Section 3.2). To realise the urn method for dynamic generation of tasks would be possible but would imply a very large amount of work. Additionally it would need an extremely fast computer for RTE.

Using the urn method yields task lists which fulfil the ideal zero DELTA values. In practice $DELTA_q = 0$ is achieved if the RTE algorithm is correctly implemented. The preparation time values and their deviations usually differ from their ideal values. This is not a consequence of the Urn Method but it is a technical hardware problem. The RTE has to submit the tasks quickly in real time according to the sequence written in the task list. But it could have delays due to insufficient CPU speed, and the speed of its data lines, etc. These influences would lengthen the actual preparation times.

For writing an ISO-type RTE the Urn Method is strongly recommended. There are no disadvantages. But it would not always be necessary to rewrite an non-urn ISO-type RTE. If, for instance, the workload is characterised by many users submitting only small tasks quickly then a large number of "samples" is produced. A measurement using a non-urn ISO-type RTE can then achieve good DELTA values. This is due to the large number of tasks that are processed in a short measurement duration. But if such a workload has additional users submitting a few "large tasks" then the use of the Urn Method would be very advantageous.

The DEMO system, contained on the CD as part of this book, uses the Urn Method including the method of Section 6.3.3 . The reader is encouraged to look into the programs to better understand how to implement the Urn Method.

6.5 Formal and detailed description of the modifications for computing the performance values

When using the Urn Method, individual RIs must be used. This implies a modification of the computation of the performance values, which has been described illustratively in Section 6.2.2 . It is now explained more formally in accordance with the presentation in Chapter 5.

6.5.1 Total throughput vector B

Because there is no common RI, it is recommended that there should be a separate logfile for each of the users (individual logfiles). Each logfile has an RI part and SR part as recommended in Sections 3.4 and 3.5 . We have to compute the throughput values separately from the RI part of each logfile, according to the individual RI. Let v be the user number, (current number within all N_{tot} users; for N_{tot} see equation (2.4) in Section 2.6). $t_{1ind}(v)$ and $t_{2ind}(v)$ are the start and end of the individual RI of the v-th user. Its duration is

$$T_{Rind}(v) = t_{2ind}(v) - t_{1ind}(v) \quad . \qquad (6.7)$$

Let $b_{ind}(v,j)$ be the number of type j tasks submitted by user v during his RI. This value can be obtained from the RI part of his individual logfile. The throughput of this user, with respect to j type tasks is

$$B_{ind}(v,j) = b_{ind}(v,j) \; / \; T_{Rind}(v) \quad . \qquad (6.8)$$

The total throughput of j type tasks is

$$B(j) = B_{ind}(1,j) + B_{ind}(2,j) + \ldots + B_{ind}(N_{tot},j) \quad . \qquad (6.9)$$

The m values $B(1),\ldots,B(m)$ are the B vector (see equation (5.2) in Section 5.1).

6.5.2 Mean execution time vector T_{ME}

As explained in Section 6.2.2.2 the computation of T_{ME} is not completely analogous to that of B. Let $t_{ETind}(v,j,1)$ be the execution time of the first j type task in the R part of user v in its individual logfile. Let $t_{ETind}(v,j,2)$ be the analogous value of the second j type task and so on. The total number of such values is $b_{ind}(v,j)$ (see Section 6.5.1). In general is $t_{ETind}(v,j,x)$ the execution time of the x-th task of the j-th task type in the RI part of the individual logfile of the v-th user. The sum of the execution times of all j type tasks is

$$
\begin{aligned}
& sum_{ET}(j) = \\
& \\
& + t_{ETind}(1,j,1) + t_{ETind}(1,j,2) + \ldots\ t_{ETind}(1,j,b_{ind}(1,j)) \\
& + t_{ETind}(2,j,1) + t_{ETind}(2,j,2) + \ldots\ t_{ETind}(2,j,b_{ind}(2,j)) \\
& + \ldots\ldots \\
& + \ldots\ldots \\
& + t_{ETind}(N_{tot},j,1) + t_{ETind}(N_{tot},j,2) + \ldots\ t_{ETind}(N_{tot},j,b_{ind}(N_{tot},j))
\end{aligned}
$$

(6.10)

Although this formula appears complex and difficult to understand, this is not really so. The variables v, j and x can be used directly as array indices and the sum can be programmed using nested loops.

Now we can compute the T_{ME} of type j tasks:

$$T_{ME}(j) = sum_{ET}(j) \ / \ (b_{ind}(1,j) + b_{ind}(2,j) + \ldots\ b_{ind}(N_{tot},j)$$

(6.11)

The m values $T_{ME}(1), \ldots, T_{ME}(m)$ are the T_{ME} vector (see equation (5.3) in Sect. 5.1).

6.5.3 Timely throughput vector E

Let $e_{ind}(v,j)$ be the number of timely executed j type tasks of user v. It can be computed (by use of the algorithm which was explained in Section 5.4.2) from the $b_{ind}(v,j)$ type j execution times in the RI part of the individual logfile of user v.

The individual number of timely executed j type tasks of user v per time unit is

$$E_{ind}(v,j) = e_{ind}(v,j) \ / \ T_{Rind}(v) \quad . \qquad (6.12)$$

This is the timely throughput of j type tasks of user v.

The sum of the $E_{ind}(v,j)$ values of all users is the total timely throughput of j type tasks:

$$E(j) = E_{ind}(1,j) + E_{ind}(2,j) + \ldots + E_{ind}(N_{tot},j) \quad . \qquad (6.13)$$

The m values $E(1), \ldots, E(m)$ are the E vector (see equation (5.4) in Section 5.1).

6.5.4 Explanations

Sections 6.5.1 to 6.5.3 concern the use of individual rating intervals. As they have to be used and programmed for the Urn Method, this is why the computation of the performance values is explained again here in more detail than in Section 6.2.2 .

But it is important to understand that individual RIs can also be used without urns. In this case the performance values have to be computed according to Sections 6.5.1 to 6.5.3 and not as described in Sections 5.2 to 5.4 .

A program for the computation of the performance values according to Sections 6.5.1 to 6.5.3 can also be used, and works correctly, without individual RIs, i.e. if all t_{1ind} times are the common time t_1 and all t_{2ind} times are the common time t_2 .

6.6 Exercises

Preparation:
a) Write down the WPS used in Exercise 1 of Section 5.5 and convert it to DEMO format.
b) Make sure that the task list generation function of DEMO is installed.
c) Generate task lists by DEMO that have the same length as in Sections 4.5 and 5.5 (i.e. 10 tasks in each of RI, StP and SR).
d) Perform the measurement.

Exercise 1: Correctness of RTE working when using urn-generated task lists

Use the recommended values in Section 2.9.2 for the criteria, i.e.
$DELTA_q$ = 0.01, i.e. "maximum 1%"
$DELTA_h$ = 0.02, i.e. "maximum 2%"
$DELTA_s$ = 0.20, i.e. "maximum 20%"
Check whether these criteria are fulfilled.

Exercise 2: Differences in P

Compute P. Compare it with the performance values of the measurement of Exercise 1, Section 5.5 (that used the same workload but not the Urn Method). Explain the differences.

Exercise 3: ISO's three overlap criteria of the individual rating intervals

Check for the measurement of Exercise 1 whether the three overlap criteria are fulfilled (see Section 6.2.3).

Exercise 4: Using a longer RI

Repeat the measurement with a 4-times longer RI. Determine the $DIFF_h$ values and compare them with those of Exercise 1.

Exercise 5: Using a more precise RTE

Repeat the measurement of Exercise 4 but use the improved "demoshell". This module is found as file
CD/DEMO-20/DEMO-sw/user-shells/demoshell-new .
Note: This module expects the "sleep" command of the operating system of your SUT to accept non-integers for argument. If this is not the case, you cannot perform Part 2 of this exercise with DEMO. ■

Solutions
For solutions see file CD/Solutions/Solutions-Section6-6.pdf .

7 Rating the measured performance values

7.1 The principle of the ISO rating

The measured performance $P = (B, T_{ME}, E)$ is a set of physical values. The user of the IP system is interested in these values but much more whether they satisfy the user entirety requirements.

Therefore ISO/IEC 14756 introduced a rating process which compares P with those performance values which are actually required by the user entity (reference values). These values are

1. the total throughput reference vector B_{Ref}
2. the mean execution time reference vector T_{Ref}
   and
3. the requirement "all tasks are completed timely" .

The result of the rating is the final decision on the SUT: "satisfactory" or "unsatisfactory".

7.2 The ISO theoretical reference machine

The ISO theoretical reference machine does not really exist. It is defined as a fictive SUT that just fulfils the timeliness functions stated in the workload (when the SUT is driven by this workload). No task will be executed faster than necessary but all tasks are executed in time. The performance of this fictive SUT represents the requirements of the user entirety and is named P_{Ref} .

$$P_{Ref} = (B_{Ref}, T_{Ref}, E_{Ref}) \qquad (7.1)$$

$$B_{Ref} = (B_{Ref}(1), B_{Ref}(2), \ldots, B_{Ref}(m)) \qquad (7.2)$$
$$T_{Ref} = (T_{Ref}(1), T_{Ref}(2), \ldots, T_{Ref}(m)) \qquad (7.3)$$
$$E_{Ref} = (E_{Ref}(1), E_{Ref}(2), \ldots, E_{Ref}(m)) \qquad (7.4)$$

P_{Ref} can be determined as follows.

Determination of T_{Ref} :
T_{Ref} is the set of required execution time mean values of m task types. These required mean execution times can be computed directly from the timeliness functions of the WPS as shown in Section 7.3.1 ; no additional data are required. This yields the values of the m components of T_{Ref} .

Determination of B_{Ref} :
Looking at the principles of user emulation by the RTE we can see that the throughput value of every task type can be computed only from the values defined in the WPS (as shown in Section 7.3.2). No additional data are required except the execution time mean

values. But these values, the m components of T_{Ref} , are already known; they also come from the WPS. This yields the values of the m components of B_{Ref}.

Determination of E_{Ref} :
It is not necessary to perform a separate computation for determing the timely throughput E_{Ref} of the theoretical reference machine. According to the third property cited in Section 7.1, the timely throughput is already known. It is identical to the reference throughput because, as stated in the definition of the theoretical reference machine, all tasks are completed timely. Therefore it is true:

$$E_{Ref} = B_{Ref} \qquad (7.5a)$$

i.e.

$$E_{Ref}(j) = B_{Ref}(j),\ j = 1, 2, \ldots, m \qquad (7.5b)$$

7.3 Computation of the reference performance values

7.3.1 Computation of T_{Ref}

In equation (7.3), T_{Ref} is the set of the terms $T_{Ref}(1), T_{Ref}(2), \ldots, T_{Ref}(m)$, where m is the number of task types. $T_{Ref}(j)$ can be computed from the timeliness function of the type j tasks. This because the theoretical reference machine executes the tasks as fast as is needed. This implies that it responds only with execution times which equal the upper time class boundaries of the timeliness functions. The computation of $T_{Ref}(j)$ will be explained using the examples of Section 2.5 .

Example 1:
We use the timeliness function TF_1 (according to Fig. 2-4 in Section 2.5). 90% of the tasks may have an execution time of 2.00 seconds. The remaining 10% of the tasks may have an execution time of 10.00 seconds. The mean value is

0.90*2.00sec + 0.10*10.00sec = 2.80 seconds .

Assuming that task type 1 uses this timeliness function we have

$T_{Ref}(1)$ = 2.80 seconds .

Example 2:
We use the timeliness function TF_2 (according to Fig. 2-5 in Section 2.5). All of the tasks may have an execution time of 3.00 seconds. This is also the mean value. Assuming that task type 2 uses this timeliness function we have

$T_{Ref}(2)$ = 3.00 seconds .

Example 3:
We use the timeliness function TF_3 (according to Fig. 2-6 in Section 2.5,). 90% of the tasks may have an execution time of 2.00 seconds. 8% of the remaining tasks may have an execution time of 10.00 seconds. The remaining 2% may have an execution time of 20.00 seconds. The mean value is

0.90*2.00sec + 0.08*10.00sec + 0.02*20.00sec = 3.00sec

Assuming that task type 3 uses this timeliness function we have

$T_{Ref}(3)$ = 3.00 seconds .

Let z be the number of time classes of the timeliness function of type j tasks. $T_{Ref}(j)$ is to be computed as follows.

$$
\begin{aligned}
T_{Ref}(j) = \quad & g_T(1) * r_T(1) \\
& + g_T(2) * (r_T(2) - r_T(1)) \\
& + g_T(3) * (r_T(3) - r_T(2)) \\
& + \ldots \\
& \vdots \\
& + \ldots \\
& + g_T(z) * (r_T(z) - r_T(z-1))
\end{aligned}
\qquad (7.6)
$$

$g_T(k)$ is the boundary of time class k of the timeliness function. $r_T(k)$ is the relative time class frequency of time class k of the timeliness function.

Which timeliness function is assigned to a task type can be seen from the task type definition of the WPS (for an example see Fig. 2-10 in Section 2.7.2). In general there are m task types in the WPS and p timeliness functions. Also $p \leq m$ because sometimes two or more task types use the same timeliness function. Performing the computation for all p timeliness functions defined in the WPS delivers all m values $T_{Ref}(j)$. This is the execution time mean value reference vector T_{Ref} (see equation (7.3)).

7.3.2 Computation of B_{Ref}

B_{Ref} is the set of the values $B_{Ref}(1), B_{Ref}(2), \ldots, B_{Ref}(m)$ where m is the number of the task types. These reference values result from the hypothetical situation that, according to Section 7.2, the SUT responds only with execution times which just satisfy the timeliness functions. No task will be executed faster than necessary, but all tasks are executed in time. Therefore the execution time mean value $T_{ME}(j)$ of the theoretical reference machine equals the corresponding mean execution time reference value $T_{Ref}(j)$. This is true for all task types.

Using this fact, it is possible to compute $B_{Ref}(j)$ using the definition of throughput as follows. Throughput is the number of tasks divided by the time in which the tasks are submitted. For this time it is a good idea to take the mean duration of a chain. Annex B.2 of the standard illustrates clearly how to derive the formula using this idea. This is no proof in a strict mathematical sense. But, because it yields the correct result, it is a good guide for programming a subroutine for computing the formula. We can then obtain, after much mathematical work, the so-called beta formula. It computes $B_{Ref}(j)$.

Note: The name of this formula is historical. It originated in the early drafts of the German Standard DIN 66273 [DIN01] (see also [DIRLE02] and [DIRLE03]), parts of which were used when writing the ISO/IEC 14756. In those German papers the Greek letter beta was used instead of B_{Ref}. ■

$$B_{Ref}(j) = \sum_{i=1}^{n} \left[\frac{N_{user}(i) \cdot \left\{ \sum_{l=1}^{u} \left[q(i,l) \cdot a(j,l) \right] \right\}}{\sum_{l=1}^{u} \left\{ q(i,l) \cdot \left[\left(\sum_{k=1}^{L_{Chain}(l)} h(i, f(l,k)) \right) + \left(\sum_{k=1}^{L_{Chain}(l)-1} \left[\mathrm{T}_{Ref}(f(l,k)) \cdot M(f(l,k+1)) \right] \right) + \mathrm{T}_{Ref}(f(l, L_{Chain}(l))) \cdot M^{*}(i) \right] \right\}} \right]$$

(7.7a)

$$M^{*}(i) = \sum_{l=1}^{u} \left[q(i,l) \cdot M(f(l,1)) \right]$$

(7.7b)

a(j,l) is the number of tasks of the *j*-th task type within a chain of the *l*-th chain type .

f (l,k) is the task type of the *k*-th task in a chain of the *l*-th chain type .

$L_{chain}(l)$ is the length (number of tasks) of a chain of the *l*-th chain type .

M(j) is the task mode of the *j*-the task type .

The computation is not explained in full here. The important fact seen from the formula is that it uses only arguments which are found in the WPS. I.e. $B_{Ref}(j)$ (and the vector B_{Ref} , see equation (7.2)) can be computed from the values found there. B_{Ref} is uniquely defined by the WPS. No additional data and no measurement are required.

Despite the formula being complex (see equations (7.7a), (7.7b)) it can easily be used for the calculation of B_{Ref} . Even for a large WPS a present day computer can execute it quickly.

Note: The author developed the beta-formula using a stochastic mathematical model and solving its so-called Chapman-Kolmogorov equations. He carried out this work for DIN in the 1980s. The formula was published in [DIN01] and appeared again in [DIRLE02], and is now incorporated on ISO/IEC 14756. A source program for computing the formula was published in [DIN01] and appeared again in [DIRLE02], is also found in Annex E of ISO/IEC 14756, and additionally a subroutine is contained in the DEMO system, found on the CD as part of this book. ■

Performing the computation of equations (7.7a,b) for all task types delivers the m values $B_{Ref}(j)$. They represent the timely throughput reference vector B_{Ref} (see equation (7.2)). E_{Ref} is then (see equation (7.5a)) computed.

7.4 Throughput rating

For the throughput rating the SUT throughput B is compared with the throughput B_{Ref} of the theoretical reference machine. The throughput rating value of type j tasks is $R_{TH}(j)$.

$$R_{TH}(j) = B(j) \ / \ B_{Ref}(j) \qquad (7.8)$$

For each of the m task types the throughput rating value has to be computed. This is the throughput rating vector R_{TH}.

$$R_{TH} = (R_{TH}(1),\ R_{TH}(2),....,R_{TH}(m)\ ) \qquad (7.9)$$

The throughput rating is simple. For each task type only two results are possible:

$R_{TH}(j) < 1$: "Less than reference" i.e. unsatisfactory or
$R_{TH}(j) \geq 1$: "Not less than reference" i.e. satisfactory .

Only if all m values $R_{TH}(j)$ are not less than 1 does the SUT satisfy the user entirety throughput requirements.

7.5 Rating the mean execution times

For the mean execution time rating the SUT mean response time vector T_{ME} is compared with the mean execution time vector T_{Ref} of the theoretical reference machine. The execution time rating value of the type j tasks is $R_{ME}(j)$.

$$R_{ME}(j) = T_{Ref}(j) / T_{ME}(j) \qquad (7.10)$$

For each of the m task types the mean execution time rating value has to be computed. This is the mean execution time rating vector R_{ME} .

$$R_{ME} = (R_{ME}(1), R_{ME}(2), ..., R_{ME}(m)) \qquad (7.11)$$

The mean execution time rating is simple. For each task type only two results are possible:
$R_{ME}(j) < 1$: "Less than reference" i.e. unsatisfactory or
$R_{ME}(j) \geq 1$: "Not less than reference" i.e. satifactory .

Only if all m values $R_{ME}(j)$ are not less than 1 does the SUT satisfy the user entirety mean execution time requirements.

Attention is drawn to the fact that in equation (7.10) above, contrary to equation (7.8) the reference value is the numerator. This was intentionally so defined to achieve the same rating principle as for the throughput: the quotient is less than 1 for "unsatisfactory" and not less 1 for "satisfactory".

7.6 Timeliness rating

Why is timeliness rating needed ? The mere rating of the mean execution times is inadequate. There could be some very long execution times that would be balanced statistically by some very short execution times. The long execution times would be unacceptable and the too short execution times of little benefit.

This was the background for the introduction into the ISO method of the timeliness functions (see Section 2.5) and the performance term "timely throughput" E (see Section 5.4).

The obvious way for a timeliness rating could be to use, analogous to the throughput rating, the quotient $E(j)/E_{Ref}(j)$ as a criterion. Or we could use the quotient $E(j)/B_{Ref}(j)$ (which means the same because of equation (7.5a)). But this criterion does not check the third requirement of Section 7.1 satisfactory.

For an explanation we consider task type 3. If the actual throughput $B(3)$ is greater than $B_{Ref}(3)$, then a curious situation could arise. There are enough short execution times for achieving $E(3) > E_{Ref}(3)$. I.e. our criterion quotient $E(3)/E_{Ref}(3)$ is not less 1, producing the decision "satisfactory". However it could happen that not all type 3 tasks fulfil the according timeliness function. This would violate the general rule that a criterion quotient of 1 means "satisfactory" (here "all tasks completed timely").

The problem is solved by taking the quotient $E_{Ref}(j)/B(j)$ as a criterion. I.e. we define the following timeliness rating term:

$$R_{TI}(j) = E(j) / B(j) \qquad (7.12)$$

Because not more than all tasks can be timely executed the term $R_{TI}(j)$ cannot be greater than 1. Therefore

$R_{TI}(j) < 1$ means "unsatisfactory timeliness"
$R_{TI}(j) = 1$ means "satisfactory timeliness"

For each of the m task types the timelines rating value $R_{TI}(j)$ has to be computed. This is the timeliness rating vector R_{TI}:

$$R_{TI} = (R_{TI}(1), R_{TI}(2), ..., R_{TI}(m)) \qquad (7.13)$$

Only if all m $R_{TI}(j)$ values equal 1 does the SUT satisfy the timeliness requirements of the user entity.

7.7 Overall rating

7.7.1 General ISO rule of rating

There are m rating values $R_{TH}(j)$ and m rating values $R_{ME}(j)$ and m rating values $R_{TI}(j)$. I.e. there are $3*m$ rating values. If at least one value is less 1 then the SUT fails to fulfil at least one of the user requirements specified in the WPS and must be rated "unsatisfactory". Otherwise the SUT is "satisfactory" and could even perform better than the requirements.

For acceptable tolerances of the rating values see Section 8.4 .

7.7.2 Extended ISO rating rule

ISO/IEC 14756 allows an extension of this simple rating. The user of the standard is free to define personal bandwidths for the R-values (by setting X-values as follows) and to accept only SUTs within such ranges.

$$X_{TH\text{-}lower} \leq R_{TH}(j) \leq X_{TH\text{-}upper} \qquad (7.14a)$$

$$X_{ME\text{-}lower} \leq R_{ME}(j) \leq X_{ME\text{-}upper} \qquad (7.14b)$$

$$X_{TI\text{-}lower} \leq R_{TI}(j) \leq 1 \qquad (7.14c)$$

One purpose of the standard in allowing such bandwidths is to make it possible to rate how much a SUT is better than required by the WPS. This, for instance, might be of interest if a too good SUT (i.e. exceeding the requirements) could justify a premium price.

Concerning the equation (7.14c) it would make no sense to have an upper limit such as "$X_{TI\text{-}upper}$"; the reason being that $R_{TI}(j)$ cannot exceed 1.

The user of the standard should be aware of the following consequence. If at least one of the values $X_{TH\text{-}lower}$, $X_{ME\text{-}lower}$, $X_{TI\text{-}lower}$ is defined as less than 1, then a SUT will be rated as "satisfactory", while the user entity will regard it as "unsatisfactory".

It is strongly recommended not to define any of the three "lower-values" as less than 1. But it could make sense, in special situations, to define $X_{TH\text{-}upper}$ and/or $X_{ME\text{-}upper}$ as greater than 1.

7.8 Exercises

Exercise 1: Determining the rating values

Consider the workload and the measured performance of solutions of Exercise 1, Section 6.6. The measurement archive is in directory

`CD/Sol-files/Mment-ch6/ARCHIVE-TTxcMm6-E1/` .

Compute the rating values.

Solutions

For solutions see file

`CD/Solutions/Solutions-Section7-8.pdf` .

8 The performance measure N_{max}

8.1 Maximum number of timely served users (N_{max})

People like to describe the performance of IP systems by a scalar value such as the old-fashioned "million instructions per second" (MIPS) of mainframe machines or "giga floating point operations per second" (GFLOPS) of super computers. Contrary to this the ISO method makes it clear that system performance measure is not a scalar but a vector (or even a set of three vectors).

To approximate the theoretical unreachable goal of a scalar performance measure we can take an ISO workload and modify it stepwise. There are many ideas of doing so.

For instance we can modify the activity types by using a replication factor (for examples see Section 11.3.1). We can perform several measurements increasing the replication factor and determine the rating values. We can use the replication factor as a performance term. Performance is the limiting value of the replication factor when the ISO rating changes from "satisfactory" to "unsatisfactory" (see Section 14.3).

One idea of modifying a workload stepwise seems to be both attractive and very practical. It is to increase the number of users of a defined workload while keeping constant all other parameters of the WPS. When the rating changes from "satisfactory" to "unsatisfactory", N_{max} is the number of users. But it is important not to change the basic characteristics of the workload when increasing the number of users. This implies that we eventually cannot increase the number of emulated users by an arbitrary increment. This aspect is detailed pointed out in Section 8.2 .

N_{max} is a powerful performance measure, but it is not an absolute measure. It always refers to a defined workload. N_{max} is a not normative performance measure of ISO/IEC 14756 but it is tolerated by the standard. It is derived from the ISO terms. It is only applicable to multi-user SUTs.

8.2 Incrementing the number of users

If the workload has only one user type (i.e. $n = 1$) the lowest possible number of users is 1. We can set $N_{user(1)} = 1$. Then $N_{tot} = 1$ and the workload is a basic workload. The smallest step for increasing N_{tot} is 1. We are free to increase N_{tot} in steps of 1 or more.

If the workload has more than one user type (i.e. $n > 1$) the situation is more complicated. As an example we take a workload with two user types, i.e. $n = 2$, each having one user. Obviously $N_{tot} = 1$ is impossible. The minimum value of N_{tot} is 2. Increasing N_{tot} from 2 to 3 users would violate the principle of keeping the basic characteristics of the workload. The relative percentages of the chain types would be changed. But increasing N_{tot} from 2 to 4 would keep the percentages the same. The minimum increment for changing N_{tot} is therefore a multiple of 2. Consequently N_{max} would be an even number.

Generally the basic workload of a given workload is found by determining the greatest common divisor N_{GCD} and then dividing all $n(i)$ by N_{GCD} .

Example: The values of the WPS are

$$N_{user}(1) = 180$$
$$N_{user}(2) = 60$$
$$N_{user}(3) = 90$$
$$\_\_\_\_\_$$
$$N_{tot} = 330$$

We find $N_{GCD} = 30$. All user numbers divided by N_{GCD} yields

$$N_{user}(1) = 6$$
$$N_{user}(2) = 2$$
$$N_{user}(3) = 3$$

These are the user numbers of the basic workload. Its N_{tot} value equals 6 + 2 + 3 = 11 . Multiplying all user numbers $N_{user(i)}$ by 1, 2, 3, etc. yields N_{tot} = 11, 22, 33, etc. The smallest possible increment (called minimum increment) of N_{tot} is 11 .

This example shows how to obtain the minimum increment and user numbers of each user type.

8.3 Measurement series

A measurement series results from using a basic workload and incrementing the user numbers stepwise by multiplying with 1, 2, 3, etc. This is a series with the minimum increment. For each step a measurement has to be performed and the rating values have to be determined. N_{max} is the greatest number of N_{tot} still having the rating result "satisfactory".

It is recommended to start a measurement series always with the lowest possible N_{tot} value, i.e. use the total user number of the basic workload for checking whether the RTE and the SUT are running well. If the measurement works correctly proceed by increasing the user number. But it is not necessary to proceed always with the minimum increment. If the rating shows excellent values some steps can be skipped. N_{tot} can be increased by a multiple of the minimum increment. When coming close to the situation in which the rating changes from "satisfactory" to "unsatisfactory" use only single steps, i.e. the minimum increment of N_{tot}. This is to determine as precisely as possible the value N_{max} .

Typically N_{max} is not exactly just a multiple of the minimum increment. For instance in the example above (in which the minimum increment is 11) it might happen that N_{tot} = 88 has a rating with all R_{TH} and R_{ME} values clearly above the limit of 1 and no R_{TI} value less than 1. And the next value (N_{tot} = 99) delivers a rating with some (or all) R values being clearly below 1 (showing "unsatisfying"). Then N_{tot} = 88 is the N_{max} value.

But in this example it is up to the person responsible to consider interpolation between 88 and 99.

Figures 8-1 and 8-2 show an example of the result of a measurement series. The workload used was similar ISO type CC2 (version "M" and REP about 20). The SUT was an Intel CPU based machine; the CPU rate was about 1.5 Mhz. The operating system was LINUX. There were m = 6 task types. Consequently each of the performance terms B, T_{ME} and E has 6 components (yielding 6 curves) as shown in Fig. 8-1.

The rating values are shown in Fig. 8-2 . For m = 6 task types each of the rating terms R_{TH}, R_{ME} and R_{TI} has 6 components (yielding 6 curves) as shown in Fig. 8-2 . The values of R_{TH} and R_{ME} can clearly be greater than 1. But those of R_{TI} cannot be greater than 1 (see explanations in Section 7.6).

N_{max} is defined when at least one of 3*6 =18 R-curves falls below 1 . This happens, in the example, with $R_{TI}(j)$ at about N_{tot} = 15. As explained above one could decide on N_{tot} = 16. This small bonus is due to d and ALPHA as be explained in the following Section 8.4.

8.4 Acceptable tolerances of N_{max}

The confidence intervals d(j) for checking the statistical significance of the measurement result can be interpreted as the maximum possible measurement error (see Section 4.3.2). This result has a probability of 1 - ALPHA of being correct. This implies that the assessment would be too strictly to take 1 as the limit of the rating values R. For instance if d were set to 5% of a regarded performance term then 1 - 0.05 = 0.95 can be accepted as the limit of R values, i.e. the limit for determining N_{max} is not 1 but 0.95 (see Section 7.7.1).

It is up to the person responsible for the measurement to use a limit less then 1. This results in a somewhat better value of N_{max} .

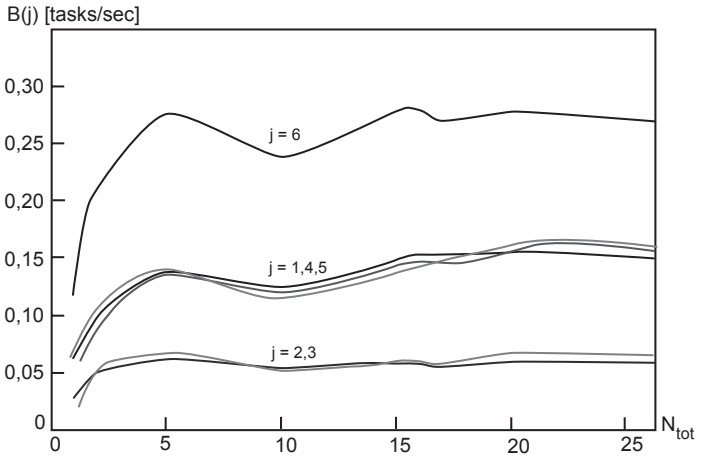
8.5 Experiences from various measurement series

With increasing N_{tot} the B curves typically rise from their initial values (from the basic workload). But at some point they stabilise and then fall sharply.

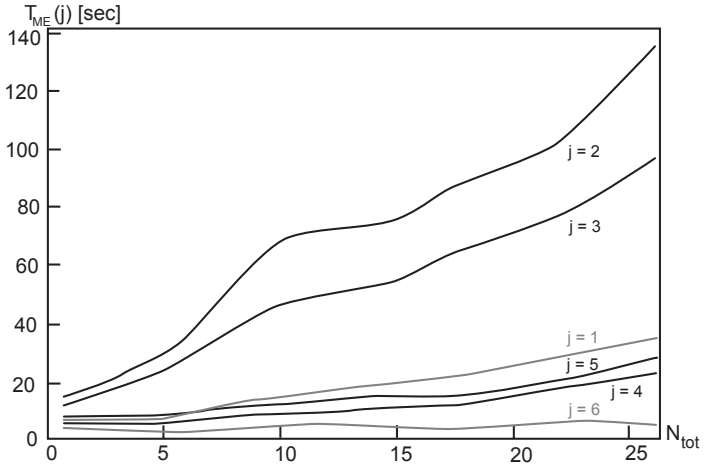
With increasing N_{tot} the T_{ME} curves typically rise from their initial values (from the basic workload). But at some point they escalate sharply.

With increasing N_{tot} the R_{TI} curves typically remain at first at their initial values (from the basic workload). These initial R_{TI} values are 1 if the SUT is fast enough compared with the timeliness requirements in the WPS of the basic workload. Elsewhere one or all initial values can be less than 1. With increasing N_{tot} suddenly one curve or some of them fall sharply.

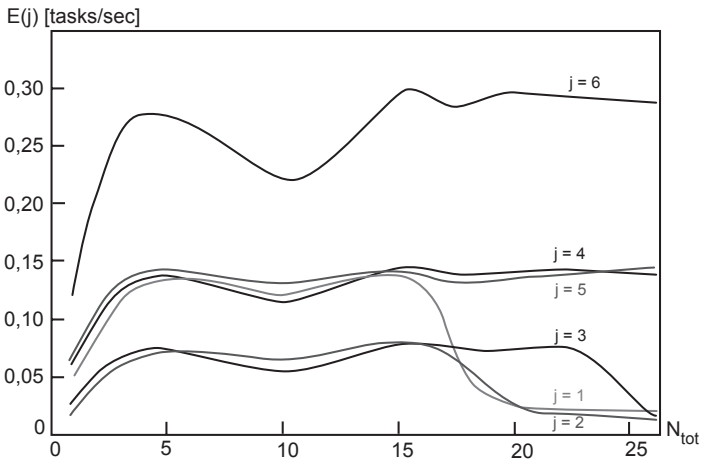
If the WPS contains no task types having the task mode value 0 (NOWAIT) then mostly the R_{TI} values are what determine the value of N_{tot} . I.e. timeliness is the deciding criterion (and not throughput or mean execution times). Examples of such workloads are the ISO workload COMPCENTER1, Version "I" and COMPCENTER2, Version "I" (see Annex F in ISO/IEC 14756 and also Sections 11.3.2, 11.3.3, 11.5.1.1 and 11.5.2.1 of this book).



8-1a Total throughput

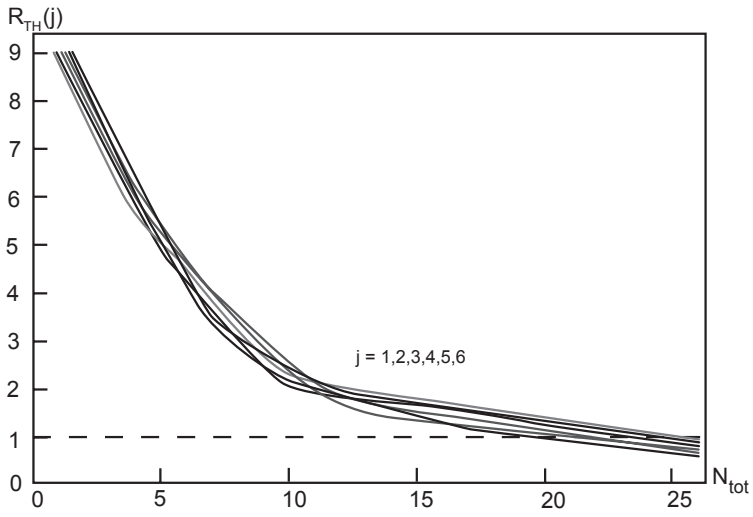


8-1b Mean execution time

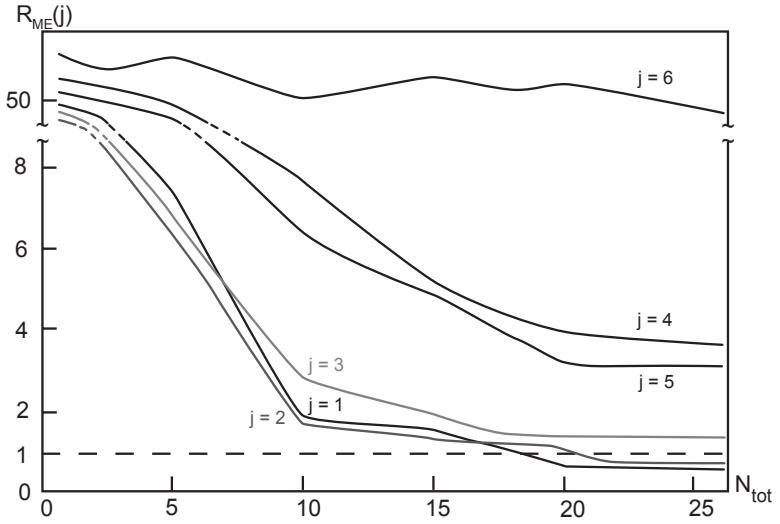


8-1c Timely throughput

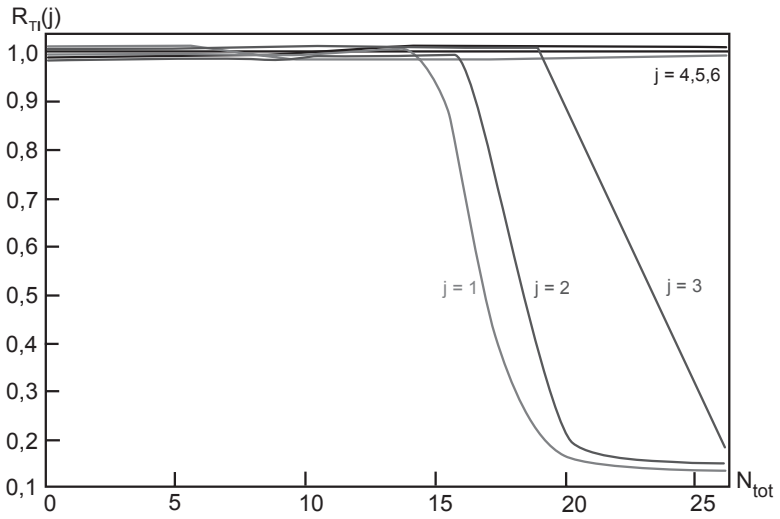
Fig. 8-1 Measured performance values of a measurement series



8-2a Throughput rating



8-2b Mean execution time rating



8-2c Timeliness rating

Fig. 8-2 Rating values of a measurement series

But if most task types defined in the WPS have the task mode value of 0 then typically the R_{TH} values are the criteria that determine the value of N_{tot} . I.e. the throughput is the deciding criterion (and not mean execution times or timeliness). Examples of such workloads are the ISO workloads COMPCENTER1, Version "B" and COMPCENTER2, Version "B" (see Annex F in ISO/IEC 14756 and also Sections 11.3.2, 11.3.3, 11.5.1.1 and 11.5.2.1 of this book).

Although in principle the duration of the rating interval T_R increases with increasing N_{tot} this effect is not very strong marked. Usually T_R is not much longer for $N_{tot} = N_{max}$ than for N_{tot} = value of the basic workload. T_R only increases sharply when N_{tot} is clearly greater than N_{max} .

8.6 Exercises

Exercise 1: Measurement using a basic workload

Preparation

Use the WPS of Exercise 1 of Section 6.6 but change the activity types to `TT1a`, `TT2a` and `TT3a` as shown below.

File `TT1a` :

```
#!/bin/sh
#
# Shell procedure TT1a
#
echo Procedure TT1a started
#
#
mkdir $$
cd $$
# Creating a directory having a unique name,
# which is the actual UNIX process number.
echo TT1.$$ > TT1-Res
# Bringing the name of the called shell
# and the actual UNIX process number - as
# an identifier of the run - to the output file.
#
cat TT1-Res ../science.c >> ../TT1.out
#
# A line for marking the begin of the grep result
#
echo XXXXXXXXXXXXXXXXX > xfile
cat xfile >> ../TT1.out
grep a ../science.c >> ../TT1.out
# copy the input file and the grep result
# to the output file
#
#
# Loop for additional CPU and I/O loading
#
anzahl=1
while [ $anzahl -le $REP ]
do
#echo $anzahl -th run through the loop
cat ../science.c >> ./TT1.intern
cp ../science.c scie-local
grep a scie-local > grep-result
cat grep-result >> ./TT1.intern
echo $anzahl > number
cat number >> ../TT1.out

anzahl=`expr "$anzahl" + 1`

#echo $anzahl in creased

done

rm -f *
cd ..
rmdir $$
# Cleaning temporary files and the directory.
#
echo End of procedure TT1a
# == End of procedure ==
```

Files `TT2a` and `TT3a` are analogous. All three files are found in directory
`CD/Sol-files/Mment-ch8/OSCPs-TTxa-shells/` .

Part 1
Write down WPS and SAG.DAT .

Part 2
Check whether this workload is a basic workload.

Part 3
Determine the increment of N_{tot} for performing a measurement series.

Part 4
Set N_{tot} = 2 and perform a measurement using REP = 1.
Report the measured performance and rating values.

Exercise 2: Performing a measurement series

Become familiar with the operator utility "./runMeasurement x" of DEMO.
Perform a measurement series and determine N_{max}. Use a REP value suitable for your SUT. Store the measurement result files using the operator utility
"./saveMment ddd" or "./saveMment2 ddd".
All files are stored in the directory "ddd" .

Note: For instance if your SUT has a 1.2 Mhz Intel CPU, a value of REP=30 can be suitable. If your SUT uses a faster CPU the following problem can arise. The maximum number of users to be emulated by DEMO 2.0 is 99. But N_{max} can be greater than 99. Then use a value greater than 30 for REP. There is another problem that can limit the maximum number of users emulated by DEMO. Most of the LINUX operating systems support less than 99 active Xterminals or remote shells when using the default values of the operating system parameters. When using DEMO in this exercise, the easiest way is to set REP to a value that does not cause N_{max} to exceed 25 or 30.

Exercise 3: Repeat the measurement series with a modified WPS

Use the same workload as in Exercise 2. But modify its WPS by setting the task modes of all task types to 0 (NO WAIT). Use the same REP value as in Exercise 2. Observe the effect of increasing N_{tot} on B and the rating values.

Exercise 4: Greater REP value

Repeat the measurement series of Exercise 2 with a significantly greater REP value.

Solutions
For solutions see file
CD/Solutions/Solutions-Section8-6.pdf .

9 Summary of the ISO system performance measurement method

The fundamentals of the ISO method have certainly been understood by reading the first eight chapters. Chapter 9 is an intermediate chapter. It is intended to help the reader by summarising the essentials. It should also give a better understanding of the basic principles explained in the previous chapters.

9.1 The steps of an ISO-type measurement run

The goal of an ISO-type measurement run is to determine the performance of an IP system, the SUT, in a well defined environment and in a reproducible way. This implies that the measurement cannot be performed with real users. It has to be performed in a test-bed that simulates the entirety of users in a realistic way (see Fig. 9-1).

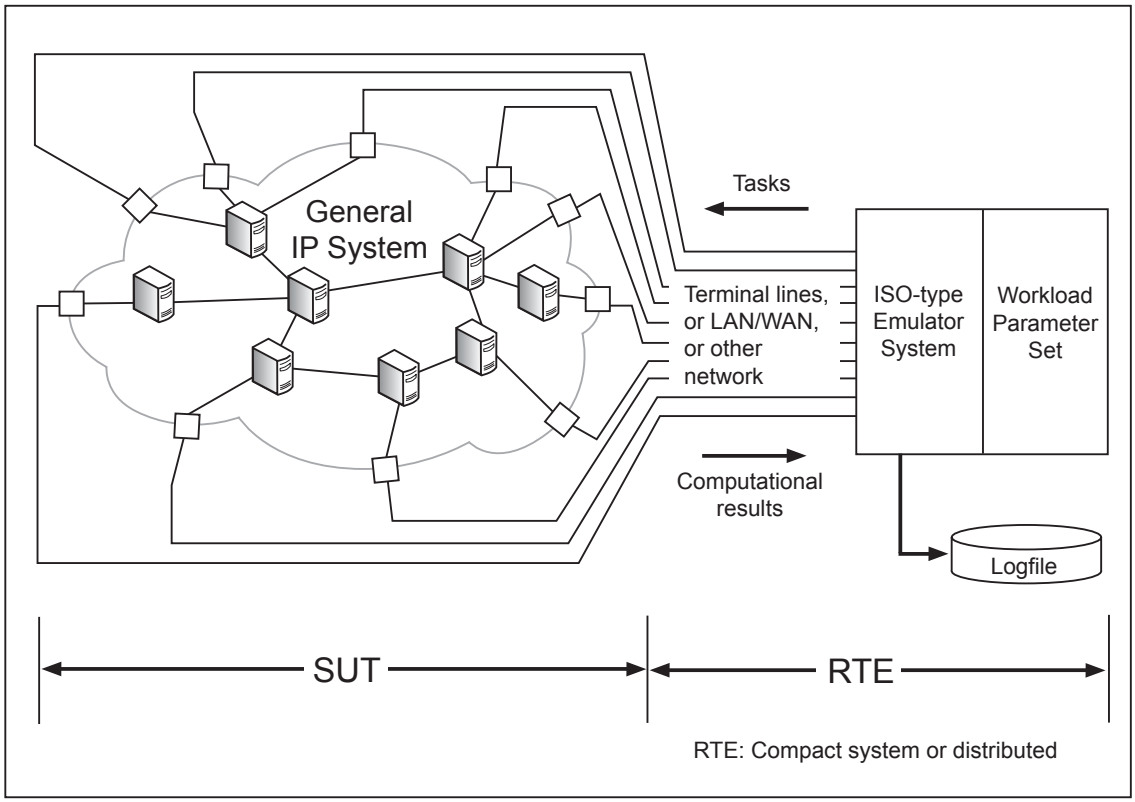


Fig 9-1 Test bed for the measurement

The steps of the measurement procedure (see Fig. 9-2) are as follows. Please notice that, for easier understanding, the sequence of steps is chosen slightly different from that in ISO/IEC 14756. But the ISO procedure remains unchanged.

9.1.1 Step 1: Specification of the workload

The workload has to be specified in accordance with the ISO workload model (see Chapter 2). For a summary of its components see Section 2.10 in Chapter 2 and Annex C in ISO/IEC 14756. Short overview of the included data and information:

- the WPS, i.e. the basic parameters of a defined user entirety and the values of all user behaviour parameters, including the user requirements for execution times of all task types
- the complete application software to be used by the user entirety together with any special system software
- the complete stored data of the applications
- For checking the correctness and the quality of the measurement have to be provided the so-called correct computation results, the so-called DELTA criterion and the requirements concerning the statistical significance.

9.1.2 Step 2: Installation of the applications on the SUT

The complete application SW has to be installed on the SUT, including the according stored data, and made ready for running in real user operation. The applications must be tested comprehensively for correct operation and computational results. Only proceed to Step 3 when Step 2 has been completed successfully.

9.1.3 Step 3: Connecting the SUT to the RTE

The SUT can now be connected to the RTE. The connection has to be implemented by real data communication lines. As above, proceed to the next step only when comprehensive check for correct function of all applications of all emulated users has been performed.

9.1.4 Step 4: Loading the RTE with the WPS

The WPS has to be loaded onto the RTE. The loading includes the set of all input strings and the input variation rules.

9.1.5 Step 5: The measurement run

Two forms of the measurement run are possible depending on the rating intervals. The basic form has common RIs while the other has individual RIs (see Section 6.2).

9.1.5.1 Basic form of measurement with common rating intervals

Define the duration of the StP and the T_R of the RI. These two values have to be set according to the experience of the person responsible for the measurement. They define the times t_1 and t_2 .

a) Stabilisation phase (StP)
Start the RTE (time t_0). At this the StP begins. Wait for the end of the StP (time t_1).

b) Rating interval (RI)
The RI begins at t_1 . At this time the recording of the logfile has to be started. If you have split the logfile into an RI and an SR part (see Sections 3.4 and 3.5) recording is on the RI part. The RI ends at t_2 .

c) Supplementary run (SR)
The SR begins at t_2, which has to be marked in the logfile. With a split of logfile, recording the RI part is now complete and recording of SR part starts. When the computational results of all tasks submitted before t_2 have been received, time t_3 is reached, the SR ends and the recording of the logfile stops. At this time the RTE can also be stopped.

d) Saving the computational results
All computational results of the SUT have to be saved unless they are being checked dynamically for computational correctness (see Sections 4.1. and 4.4).

9.1.5.2 Measurement with individual rating intervals

Define the planned durations of the StP and T_R of the RI. These two values have to be set according to the experience of the person responsible for the measurement. They define the times t_1 and t_2 .

a) Stabilisation phase (StP)
Start the RTE (time t_0). At this time the StP begins. Wait for the planned end of the StP (time t_1). Check the task chain of each user and wait for its completion. This moment is t_{1ind} and the start of the individual RI of the user. Do not forget to repeat for each user. For details see Section 6.2.1 .

b) Rating interval (RI)
The individual RI of each user starts at his t_{1ind} . At this time the recording of each individual logfile has to be started. With a split logfile the recording is on the RI part (see Sections 3.4 and 3.5). If you are using the Urn Method do not choose any chain start for t_{1ind}, but wait for the start of a new cycle. For "cycle" see Section 6.3.2.1 . The planned end of the RI is t_2 . Check the task chain of each user and wait for its completion. This time is t_{2ind} and the end of the individual RI of the user (see Section 6.2.1) If you are using the Urn Method, check the chain cycle and wait for its completion. This time is t_{2ind} and the end of the individual RI. Do not forget to repeat for each user.

c) Supplementary run (SR)
The individual SR of each user begins at his t_{2ind} which has to be marked in the individual logfile of each user. With a split logfile (see Sections 3.4 and 3.5) the individual RI part is now complete and recording the individual SR part starts. When the computational results of all tasks submitted before t_{2ind} have been received, time t_{3ind} is reached and the individual SR ends. When all users have reached their t_{3ind} , this common time is t_4 and the recording of the individual logfiles stops. At this time (or later) the RTE can also be stopped.

d) Saving the computational results
All computational results of the SUT have to be saved unless they are being checked dynamically for computational correctness (see Sections 4.1 and 4.4).

9.1.6 Step 6: Checking the correct working of the SUT

All computational results of each user have to be checked against the correct results stored in the workload (see Section 4.1). Clearly, this checking is laborious and needs suitable computing capacity. With input variation it is even more laborious (see Section 4.4). If the checking fails the measurement is not valid. The SUT has to be debugged and a new measurement has to be planned for the improved SUT.

9.1.7 Step 7: Checking the correct working of the RTE

This step checks the RTE against the `DELTA` values specified in the workload (see Section 4.2). Such criteria are for the relative chain frequencies, the means and the standard deviations of the preparation times. The corresponding measured values have to be computed from the logfiles. None of them may differ more than allowed by the `DELTA` value. They have to be checked for both the RI and the SR parts of the logfiles. If there are differences greater than allowed in the workload, the RTE has functioned inadequately and the measurement is not valid. The RTE has to be corrected and new measurement has to be planned with the improved RTE.

9.1.8 Step 8: Checking the statistical significance and the RI overlap

As explained in Section 4.3 the significance of the measured results must be checked. This check is using the RI part of the logfile. The sequential test of the execution time mean values has to be done separately for all m task types. The test criteria are the confidence coefficient `ALPHA` and the confidence intervals `d` as specified in the workload. If any result was "NOT OK", the measurement was unsuccessful. The statistic uncertainty of the measured preparation time mean value of at least one task type was less the requirement in the workload and the reason must be investigated. A new measurement with improved test conditions (for instance a longer StP and/or longer RI) has to be planned.

With using individual RIs (see Section 6.2.1) an additional check is necessary. As explained in Section 6.2.4 the individual RIs must have sufficient overlap. The ISO criteria are listed in Section 6.2.4. If any of these three criteria are not fulfilled the measurement is invalid. The reasons have to be investigated and a new measurement with improved test conditions has to be planned.

9.1.9 Step 9: Calculating the performance values

Calculate the performance values,

i.e. $P = (B, T_{ME}, E)$

from the RI part of the logfile (see Section 9.2.1).

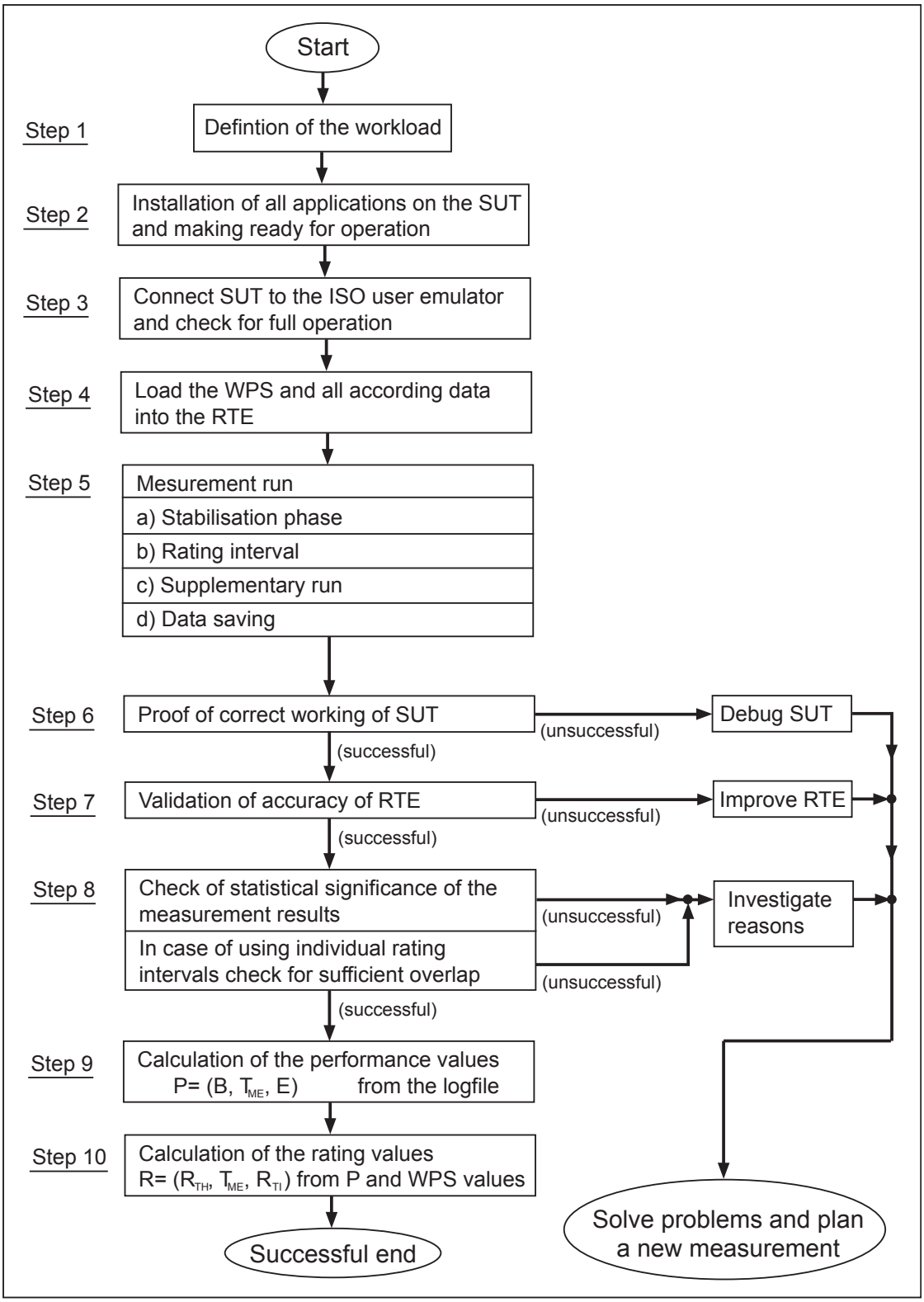


Fig. 9-2 The steps of the ISO measurement procedure

9.1.10 Step 10: Calculating the rating values

Calculate the rating values,

i. e. $R = (R_{TH}, R_{ME}, R_{TI})$

by comparing the performance values (i.e. P) to the reference performance values (i.e. P_{Ref}) which are derived from the users requirements defined in the WPS (see Section 9.2.2).

Both Steps 9 and 10 are explained further in Section 9.2 .

9.2 Computing the measurement results

9.2.1 Calculation of the performance values

This calculation (Step 9, see Section 9.1.9) is only meaningful if each of the preceding Steps 6, 7, 8 had delivered the result "satisfactory". Otherwise the calculation can be omitted because the computed performance values have no relevance.

The calculation needs the following data (see Chapter 5):

a) Data from the measurement run
- the values of t_1 and t_2 (start and end of the RI)
  or, if individual RIs were used, the values
  of t_{1ind} and t_{2ind} of all users
- the logfile (containing the recorded data sets of all tasks submitted to the SUT within the RIs)

b) Data from the workload
- the WPS of the workload (which contains the important timeliness functions)

The numbers $b(j)$ of executed tasks submitted within the RI can be computed from the logfile data. Hereby the throughput values $B(j)$ are computed yielding the throughput vector B (see Section 5.2).

The execution times $T_{ME}(j)$ can also be computed from the logfile, yielding mean execution time vector T_{ME} (see Section 5.3).

The numbers $e(j)$ of timely executed tasks can be computed from the logfile data, together with the values of the timeliness functions. Hereby the timely throughput values $E(j)$ are computed, yielding the timely throughput vector E (see Section 5.4).

The measured performance $P = (B, T_{ME}, E)$ of the SUT, with respect to the used workload, is determined. This is a set of $3*m$ values, where m is the total number of task types which were defined in the workload:

$$P = \begin{bmatrix} B(1), & B(2), \ldots, & B(m) \\ T_{ME}(1), & T_{ME}(2), \ldots, & T_{ME}(m) \\ E(1), & E(2), \ldots, & E(m) \end{bmatrix} \qquad (9.1)$$

If individual RIs are used some other data from the measurement are needed (see Section 6.2.1). Instead of one pair of values (t_1 and t_2), the measurement delivers N_{tot} pairs of values (t_{1ind} and t_{2ind}) where N_{tot} is the total number of users. The computation of P has not to be performed according to Sections 5.2 to 5.4 but to Sections 6.2.2 and 6.5.1 to 6.5.3. Especially if applying the Urn Method the individual RIs have to be used and the performance values have to be computed according to this method.

9.2.2 Calculation of the rating values

This calculation (Step 10, see Section 9.1.10) is only meaningful if each of the preceding Steps 6, 7, 8 had delivered as a result "satisfactory". Otherwise the calculation can be omitted because the computed rating values have no relevance.

The calculation needs the following data (see Chapter 7):

a) The performance data from the measurement run
(i.e. the $3*m$ performance values of P as computed according to Section 9.2.1)

b) The WPS from the workload

The performance reference values of the ISO theoretical reference machine (see Section 7.2) have to be computed from the WPS as follows.

According to Section 7.3.2 the throughput reference vector

$$B_{Ref} = (B_{Ref}(1), B_{Ref}(2), \ldots, B_{Ref}(m))$$

and according to Section 7.3.1 the mean execution time reference vector

$$T_{Ref} = (T_{Ref}(1), T_{Ref}(2), \ldots, T_{Ref}(m))$$

have to be computed from the WPS values. From the measured performance values and the reference values, the three rating vectors are now computed:

- Throughput rating vector (according to Section 7.4)
$$R_{TH} = (R_{TH}(1), R_{TH}(2), \ldots, R_{TH}(m))$$
- Mean execution time rating vector (according to Section 7.5)
$$R_{ME} = (R_{ME}(1), R_{ME}(2), \ldots, R_{ME}(m))$$
- Timeliness rating vector (according to Section 7.6)
$$R_{TH} = (R_{TH}(1), R_{TH}(2), \ldots, R_{TH}(m))$$

The total rating uses these 3*m rating values as described in Section 7.7.1. Generally, if none of these 3*m values is less than 1 then the SUT performance is satisfactory. If any R value is less than 1 then the SUT performance is unsatisfactory with respect to the requirements of the user entirety defined in the workload.

For special aspects of the total rating see Sections 7.7.2 and 8.4 .

9.3 The measurement report

The following principles are not part of the ISO/IEC 14756, but are recommended for the measurement report. ISO did not define form and contents of a measurement report.

9.3.1 Principles

9.3.1.1 Completeness

The measurement report should describe all facts, data and steps of a measurement. It has to be comprehensive; no essentials with respect the ISO method may be omitted.

9.3.1.2 Detailed Report

The measurement report should give a detailed description of all facts, data and steps performed.

9.1.3.3 Clarity

The representation of the facts should be unambiguous and clear. The use of data processing jargon should be reduced to the unavoidable minimum. Do not use the jargon of companies, manufacturers, service groups, etc. Add a complete list of used acronyms and abbreviations with clear explanations.

9.1.3.4 Data formats

These data formats should be used in the report:

a) Formats readable by humans
There are parts intended to be read by a human reader (text in the widest sense). Text, including graphics, tables, etc. should be stored in a long-lasting machine independent format, readable a lot of years later. Do not use the format of a short-lived text processing system or a local archive. Recommended are formats such as PDF (Adobe), PS (Postscript) or TEX or LATEX. Additionally produce a hardcopy printout and a COM (computer output on micro film) of the most important parts.

b) Formats generally machine-readable
There are parts of the report which are primarily intended to be read by machines (as, for instance, data files, programs) and possibly also by humans. Use for those parts as far as possible a data format which is independent of operating systems, application systems, database systems, etc. It is recommended that ASCII text be used wherever possible especially for programs and related information, such as operating system command procedures, control data sets etc.

c) Formats specifically machine-readable
These are special data which are oriented to a SUT machine type and/or its system software, which are not intended to be read by a human. They can only be stored in an machine-specific format. To provide for future usage, it is recommended that these data be converted to a less machine-dependent format and additionally stored.

9.3.1.5 The storage medium

The storage medium should be such that the report is completely readable long after its creation. Therefore do not use magnetic tapes or discs. The readers can change quickly. Operating systems supporting them change even quicker. Practical experience is that such media become unreadable after a few years. It is recommended that 4.8 inch CD-ROMs be used, written in ISO9660 format.

9.3.1.6 Reproducibility

The report should be written in such a manner that the measurement can be repeated using any suitable ISO-type RTE by any personnel having sufficient specialised knowledge of performance measurement and of the ISO method even if they were not involved in the original measurement itself.

The purpose of the above recommendations (Sections 9.3.1.1 to 9.3.1.6) is to create a report which allows everyone, now and in the future to have full information on the measurements performed and to be able to repeat the measurements with new personnel. This should be possible independent of the availability of the persons who carried out the measurements and independent of the measurement system, provided it fulfils the ISO/IEC 14756.

9.3.2 Suggested contents of the measurement report

Although ISO/IEC 14756 does not specify a list of contents for a measurement report, this can be derived to some extent from the standard. However this content list is not obvious to those unfamiliar with the standard. The following incomplete list of important parts is therefore presented as a guide. It should be supplemented as required for an actual measurement report.

Contents of the measurement report:

Part 1: Purpose of the measurement
This part is a short summary of what the purpose of the measurement, when, where and by whom it was carried it out, the names of the main responsible persons, and any other observations.

Part 2: Description of the SUT
This part includes

- detailed and complete specification of the hardware configuration and architecture of all components
- detailed and complete specification of the software configuration, including the operating system, application software and special components that are not part of the workload

- detailed and complete specification of the communication network and its configuration
- listing of all settings of system parameters of the operating system, of system SW components and of parameters of application software components

Part 3: The workload
This part contains the complete workload definition, as explained in Chapter 2. For a summary see Section 2.10. The workload should be presented in machine-readable form. The use of a CD-ROM is strongly recommended (see Section 9.3.1.5). A special storage medium may be necessary, for instance with large data file systems, complete databases or unusually large software packages.

The file structure of the workload should be compatible to the ISO workload examples of Annex F of the ISO/IEC 14756. Copies of the ISO examples are (with permission of ISO) found in Annex A of this book, see the CD which is part of this book. The ISO directory structure of the workloads is explained in Section 11.1 .

Part 4: The RTE description
For clarity, the RTE should be described completely. Data to be noted include:

- identification of the RTE software system (manufacturer, type, release, version,...)
- Information about any previous certification of the RTE, or assurance that it fulfils the specification of ISO/IEC 14756
- description of the RTE hardware configuration (description of all computers, their manufacturers, type, configuration, communication network and connections, the settings of the main system parameters, etc.).

Part 5: Measurements performed
This part lists all measurement runs which were performed. The list has an entry for each measurement run. Each entry contains at least the following information:

- Sequence number of the run
- Name of workload used
- In case of measurement series the values of the variable parameter
- Reference to the storage place, on the storage medium, of the measurement operator's protocol (see Section 9.4.1)
- reference to the storage place, on the storage medium, of the measurement results
- short report of general aspects of the measurement, e.g.
  x) "normal course" or irregularities (descriptions)
  x) problems and difficulties
  x) summary of results and comments (for instance "received expected performance values", "surprising results")
- adjustments of the main operating system parameters (for instance maximum possible number of parallel processes, maximum number of active files, partitioning, maximum number of served users,...
- system and application software settings if these are not already noted in the measurement operator's protocol.

Part 6: Measurement results
This part contains one directory for each measurement. The directories include, at least, the following information:

- sequence number of the measurement (as defined in part 5)
- reference to the storage place (on the storage medium) of the logfiles
- the measured values for and the results of the validations of:
  x) correctness of the computational results of the SUT (see Section 9.1.6, Step 6)
  x) correctness of the work of the RTE (see Section 9.1.7, Step 7)
  x) statistical significance of the measured results (see Section 9.1.8, Step 8)
  x) fulfillment of the criteria of "sufficient overlap" when using individual RIs (see Sections 6.2.4 and 9.1.8, Step 8)
- measured performance values (i.e. the three m-tupels B, T_{ME}, E)
- rating values (i. e. the three m-tupels R_{TH}, R_{ME}, R_{TI})

For a measurement series an additional directory should show

- P-curves (i. e. the set of 3*m curves of B, T_{ME} and E)
  and
- R-curves (i. e. the set of 3*m curves of R_{TH}, R_{ME} and R_{TI})
  and
- final results (for instance the value of N_{max}).

Part 7: Optional information
Here could be an auditing report, if there was any auditing voluntarily performed. Please note that the ISO/IEC 14756 does not require any auditing because it is a purely technical standard.

Part 8: Summary of the measurement project
This part is a summary of the work performed and of the main results.

Appendices (Documentation, history, comprehensive data,....)

- Annex A
  x) Measurement operator protocols
  x) Collection of measured data
- Annex B (archive)
  This Annex is the storage media itself (for instance CD-ROM). Alternatively it could be a detailed specification of the storage of all data cited in the report (for instance data archive of the measurement laboratory).

9.4 Recommendation for the documentation of a measurement run

9.4.1 Measurement operator's protocol

The 10 steps of a measurement run were explained in the Sections 9.1 and 9.2 . The measurement procedure cannot be fully automated because many unexpected events can happen. The sequence of the steps has to be controlled by an operator. These events have to be documented (on paper or via a computer screen showing a display mask). The protocol should contain the following parts.

Part 1 Header: "Performance measurement according to ISO/IEC 14756"
Date, time, operator's name, sequence number of the measurement run
Part 2 RTE: Identification (software system, hardware)
Part 3 SUT: Identification (full details)
Part 4: Chronological list of the operators actions when executing the 10 steps, notes of all expected and unexpected events.
Part 5: Detailed information on archiving (all relevant documentation and data, see Section 9.4.2)

The form of such a protocol is very dependent on the type of the RTE.

9.4.2 Archiving the measurement files

All data representing an essential part of an ISO-type measurement run should be archived. This is not a requirement of the ISO standard but it is a natural course of action. Storing the data can be done by any computer system. It would be possible for the SUT to do it; but this is not recommended. A better solution is to store the data in a database of the RTE or of a separate system. For each measurement run the following data should be stored.

1. The actual WPS used
2. The set of logfiles of all emulated users; the values t_0, t_1, t_2, t_3;
   when using individual RIs the values t_{1ind}, t_{2ind} of all users and the value of t_4 .
3. The complete set of computational results of all tasks of all users of the OP (observation period). When the person responsible for planning the measurement decided to use dynamic correctness check during the measurement run then have the results of this check to be stored.
4. The data and results of checking the correctness of the RTE's work of the OP.
5. The data and results of checking the statistical significance of the RI or for individual RIs the data and results of checking "sufficient overlap".
6. The measured performance P (3*m values) and the rating values (3*m values).

The list of names of files and directories to be archived depends strongly on the SUT and on the type of the RTE used.

9.4.3 Safekeeping period

Contrary to the measurement report, for which a very long safekeeping period is typical, the safekeeping period of the archived measurement data (see Section 9.4.2) can be shorter and is up to the decision of the laboratory. Typically the data are only needed until the measurement report is written. If an auditing procedure is intended (see Section 9.3.2, Part 7) then the data should be kept until the auditing report is finished.

9.5 Reproducibility of measurement results

When repeating a measurement the ISO-type RTE intentionally does not reproduce the user's actions in full detail. Instead it repeats the emulation using new random values. For instance it varies the preparation times and sequences of task chains. But the RTE performs the repetition by retaining the defined statistical values. Those are for instance the preparation time mean values or the relative task chain frequencies. The RTE retains it within the deviations which are defined in the WPS (DELTA values). This ensures that the final measurement results (performance values, rating values) differ only slightly when repeating a measurement.

The differences are random values. They depend on the stated values of the statistical significance ALPHA and d. The smaller these two values, the smaller are the differences of measured performance P and of the rating values when repeating a measurement. It is possible that a repeated measurement can deliver an unexpected difference from the preceding measurement. This is always possible. But experience is that for "good" ALPHA and d values the measured performance and rating values very rarely differ more than a few percent. An example of a "good pair" of values is "ALPHA=0.10" and "d=10% of the mean values" (see Section 2.9.3). Therefore the reproducibility of ISO-type measurements will be seen as very stable. If using, for instance not P but the N_{max} term as a final performance measure, then the differences tend to be even smaller.

9.6 Exercises

Exercise 1: Differences in P and rating values when repeating a measurement

Perform twice the following measurement: Use the SUT, workload and replication factor from Exercise 2 of Section 8.6 . For N_{tot} use a value that equals about N_{max} , estimated in that exercise. Take the values for ALHPHA and d_{rel} from the same exercise (i. e. 0.20 and 0.25). Set the duration T_R of the RI to about 200 seconds.

Part 1
Compare the measured performance P and the rating values of the two measurements.

Part 2
Determine for the two measurements those values of ALPHA and d_{rel} for which the statistical significance still holds.

Exercise 2: Co-relation of ALPHA and d_{rel} on the reproducibility of measurement results

Repeat the two measurement runs of Exercise 1 above, but set T_R to about 500 seconds.

Part 1
Compare the measured performance P and the rating values of the two measurements.

Part 2
Determine for the two measurements those values of ALPHA and d_{rel} for which the statistical significance still holds.

Exercise 3: Difference of N_{max} when repeating a measurement series

Perform twice a measurement series using the workload of Exercise 1 and T_R about 200 seconds.

Part 1
Compute the difference of the two measured N_{max} values.

Part 2
Determine, for each measurement of the series, those values of ALPHA and d_{rel} for which the statistical significance still holds. Then determine N_{max} in consideration of the "possible measurement error".

Solutions
For solutions see file
CD/Solutions/Solutions-Section9-6.pdf .

10 Measurement of software (run time) efficiency

Software efficiency is concerned with several qualities such as storage usage, changeability, maintainability and also with run time. Regarding software efficiency ISO/IEC 14756 deals only with run time.

10.1 A hierarchical model for information processing systems

Colloquially people used to assign the attribute speed not only to hardware but also to software. But software does not have an attribute speed. It is a sequence of information processing steps to be performed. There is no specification of how fast the steps have to be executed. In reality the execution time of each step depends on the processor (CPU), i.e. depends on the hardware. In fact we cannot define or measure the speed of software.

To find a suitable measure or term for the aspect under consideration we use a simplified model of IP systems. This is the hierarchical model presented in Fig. 10-1. In this model each level receives tasks from the level above. Each level performs received tasks by

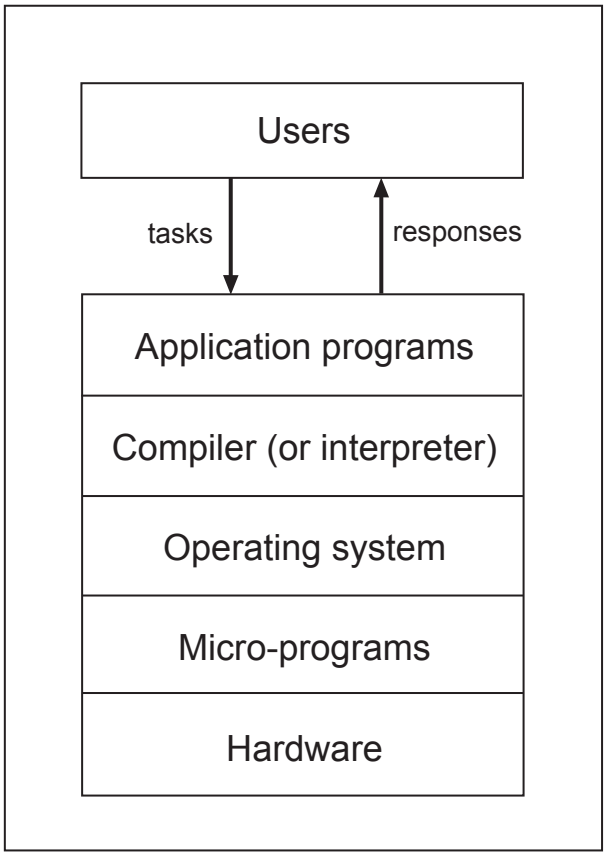


Fig. 10-1 Simplified hierarchical model of IP systems

decomposing them into subtasks and delegating these to the level below. The lowest level is the hardware and is the only one that processes its tasks by itself. It determines the execution speed of its tasks. No other level can do this.

Regarding the second level from top there is clearly a functional difference between a compiler and an interpreter; but this difference is not relevant here.

10.2 The reference environment and the term run time efficiency

Performance (in the sense of speed of execution) can only be defined for a complete hierarchy, which includes the lowest level, i.e. the hardware. If the hardware is not included, the speed of execution is undefined and a term performance cannot be defined.

Instead, a term run time efficiency can be defined as follows:

- use a hierarchy similar to that presented in Fig. 10-1;
- measure the performance P of the IP system;
- replace the implementation of the regarded level by another implementation which has the same functionality and the same upper and lower interfaces;
- measure the performance `P` of the modified IP system;
- calculate the difference in `P`.

The difference in `P` yields a measure of how more or less efficiently the second implementation transforms the tasks submitted from the upper level into subtasks submitted to the lower level compared with the first implementation. The measure is the software run time efficiency of the second implementation.

A consequence of this idea is that software run time efficiency is a measure which always refers to a reference environment. This environment consists of a defined implementation of all other levels. It includes the hardware and the user entirety. And it also includes a reference software for the level being measured.

This idea is described in Section 8.2 of the ISO/IEC 14756. Although the standard does not explicitly state that the user entirety is part of the reference environment, this is clearly so. Without a user entirety which creates and submits tasks, P values cannot be measured at all. Efficiency values could not be determined.

10.3 The measurement procedure and measures of software run time efficiency

10.3.1 The measurement procedure

The determination of the software run time efficiency of a regarded software, for instance an application program, needs two performance measurements. The procedure consists of the following steps.

Step 1
List the levels of the reference environment. Define an ISO-type workload. Define the hardware. Define the implementation of all levels except the level being measured. This yields the test bed.

Step 2
Define the reference software for the measurement level.

Step 3
Install the reference software on the test bed. This yields the reference system IP_0. Perform an ISO-type measurement. The measured performance is called P_0.

Step 4
Deinstall the reference software and install the actual software on the test bed. This yields the system IP_1. Perform an ISO-type measurement. The measured performance is called P_1.

Step 5
Compare P_1 with P_0. This yields run time efficiency values.

Terms for the run time efficiency are explained in Sections 10.3.2 and 10.3.3 .

The test bed cannot be chosen arbitrarily. Software is generally designed for an IP system of a defined type, architecture and size that are assumed to be available to the user. The IP system to be used greatly influences the software design. Software is always designed for a particular hardware configuration, its system software components and a planned group of users. Therefore the test bed has to be chosen to best represent the real environment for which the software was defined.

10.3.2 Run time efficiency terms related to task types

ISO/IEC 14756 does not define nor describe software efficiency terms explicitly. Neither does it explain how to compare the pair of measured performance values P_0 and P_1, that represent the primary measurement result. This comparison is now described here.

The measured values are

$$P_0 = (B_0,\ T_{ME0},\ E_0) \quad \text{and} \qquad (10.1)$$
$$P_1 = (B_1,\ T_{ME1},\ E_1) \qquad (10.2)$$

P_0 represents $3*m$ values.

$$B_0 = (\ B_0(1),\ B_0(2),\ldots,\ B_0(m)\ ) \qquad (10.3a)$$
$$T_{ME0} = (\ T_{ME0}(1),\ T_{ME0}(2),\ldots,T_{ME0}(m)) \qquad (10.3b)$$
$$E_0 = (\ E_0(1),\ E_0(2),\ldots,\ E_0(m)\ ) \qquad (10.3c)$$

P_1 represents $3*m$ values.

$$B_1 = (\ B_1(1),\ B_1(2),\ldots,\ B_1(m)\ ) \qquad (10.4a)$$
$$T_{ME1} = (\ T_{ME1}(1),\ T_{ME1}(2),\ldots,T_{ME1}(m)) \qquad (10.4b)$$
$$E_1 = (\ E_1(1),\ E_1(2),\ldots,\ E_1(m)\ ) \qquad (10.4c)$$

The efficiency values are calculated by dividing each P_1 component by the reference value, i.e. by the according P_0 component.

The throughput efficiency value of the type j tasks is

$$I_{TH}(j) = B_1(j) / B_0(j) \quad . \qquad (10.5)$$

The mean execution time efficiency value of the type j tasks is

$$I_{ME}(j) = T_{ME0}(j) / T_{ME1}(j) \quad . \qquad (10.6)$$

The timeliness efficiency value of the type j tasks is

$$I_{TI}(j) = E_1(j) / E_0(j) \quad . \qquad (10.7)$$

Each of these $3*m$ values indicates better efficiency if it is greater than 1, the same efficiency if it is equal to 1 and worse efficiency if it is less than 1.

In a simple example let m, the number of task types, be 3. We consider only three of the $3*m=9$ I-values. Suppose that

$$I_{TH}(2)<1 \quad \text{and} \quad I_{ME}(3)<1 \quad \text{and} \quad I_{TI}(1)>1 \quad .$$

This means that the actual software, compared with the reference software, has
- less (total) throughput for tasks of type 2;
- a greater mean execution time for tasks of type 3;
- and more timely throughput for tasks of type 1 .

This example shows that the actual software, compared with the reference software, is more efficient for (at least) one criterion and less efficient for (at least) 2 criteria. 6 of the 9 criteria are not considered in this example.

> General rule:
> Only if all 3*m I-values are not less 1
> then the actual SW is generally not less efficient,
> or even more efficient, than the reference SW.

If at least some of the $3*m$ I values are greater than 1, the actual software is more efficient than the reference software, but only for these criteria. This comparison often arises in connection with new releases of application software or operating systems. It is important to check whether a new release is at least as efficient as the previous release.

The set of $3*m$ efficiency values refers not only to the reference software but also to all other components of the reference environment. Consequently the efficiency values depend on the user entirety. If the user entirety changes, for instance by changing the

number of users, some or all 3*m efficiency values can change. Consequently one measurement is usually not sufficient for a comprehensive efficiency analysis. It is necessary to design a set of reference environments by modifying the user entirety and defining their workloads. But in most cases not all these workloads have to be completely redesigned. It can be sufficient to modify only the WPS. Additional modifications of the reference environment can be necessary. For instance, the hardware configuration or the operating system release may have to be varied.

10.3.3 A software run time efficiency term related to the performance measure N_{max}

In many cases detailed information resulting from the 3*m I values is not needed. Instead, more global information is required. It can be helpful to use the performance measure N_{max} (see Chapter 8). N_{max} is the maximum number of timely served users. Using this performance measure the software efficiency measurement procedure has the same 5 steps as described in Section 10.3.1. But steps 3 and 4 need to be modified. The single measurement run has to be replaced by a measurement series (see Section 8.3). Step 3 gives the performance value N_{max0} and step 4 gives N_{max1} . The comparison in step 5 is simple. The efficiency value $I_{maxuser}$ results from dividing the N_{max} values.

$$I_{maxuser} = N_{max1} / N_{max0} \qquad (10.8)$$

If for instance $I_{maxuser}$ equals 0.85 the actual software has a lower efficiency than the reference software. This means that 15% of the users have to be logged off in order to serve the remaining 85% timely. Or, if $I_{maxuser}$ equals 1.20 the actual software has a higher efficiency than the reference software. 20% more users can be served timely.

Understanding results like these is important when marketing new releases of operating systems or application software.

10.3.4 comparison of the two methods

There are two main differences between the methods of Sections 10.3.2 and 10.3.3 .

- Number of components of the efficiency measure:
  The efficiency measure according to Section 10.3.3, $I_{maxuser}$,
  consists of only one value. It is a scalar and easy to understand.
  The measure according to Section 10.3.2 contains 3*m I-values.
  It is more complex and produces very detailed information.

- Measurement costs and manpower:
  The measurement according to Section 10.3.3 needs two measurement series (using the same basic workload and varying the number of users). The measurement according to Section 10.3.2 needs only two measurement runs (using the same workload). The cost is less. But, see last paragraph section 10.3.2, one pair of measurements may not be sufficient for a detailed efficiency analysis.

10.4 Examples

10.4.1 Example 1: Application software efficiency

In this example there are two implementations of the application software, APS0 and APS1, which have the same functionality. APS1 is the software to be measured and APS0 is the reference SW.

10.4.1.1 The measurement environment

The test bed for application software efficiency measurement consists of the components as shown in Fig. 10-2 . The user entirety consists of 30 users. The workload has 3 task types. Further details of the workload are unnecessary for this example and for brevity are omitted.

- User entirety (example: 30 users)
- Place-keeper of application software
- Compilers and system software
- Operating system
- Hardware

Fig. 10-2 Test bed for application software efficiency measurement

Installing the application software APS0 into the test bed produces the reference IP system IP_0. Changing APS0 to APS1 produces the IP system IP_1.

10.4.1.2 The task-oriented software efficiency values

This section deals with the efficiency values according to Section 10.3.2.

An ISO-type performance measurement using IP_0 is performed. The measured performance is $P_0 = (B_0, T_{ME0}, E_0)$ as shown in Fig. 10-3 .

Task Type	$B_0(j)$ tasks/sec	$T_{ME0}(j)$ sec	$E_0(j)$ tasks/sec
1	1.61	0.81	1.61
2	0.85	2.52	0.85
3	0.81	2.03	0.81

Fig. 10-3 The performance of the reference system IP_0
(i.e. using the reference application software)

The reference application SW APS0 is then changed to the actual software APS1 which is being tested for efficiency. Compilers, system software, operating system and hardware all remain the same. This produces the system IP_1. An ISO-type performance measurement is

performed on IP_1. The measured performance is $P_1 = (B_1, T_{ME1}, E_1)$ as shown in Fig. 10-4 .

Task Type	$B_1(j)$ tasks/sec	$T_{ME1}(j)$ sec	$E_1(j)$ tasks/sec
1	1.02	2.53	0.50
2	0.41	5.02	0.35
3	0.61	9.01	0.60

Fig. 10-4 The performance of the system IP_1
(i.e. using the actual application SW)

The software efficiency values of APS1 result from the pair of measured performance values P_0 and P_1. The computed values (see equations (10.5) to (10.7) in Section 10.3.2) are shown in Fig. 10-5 .

```
Throughput efficiency:

Task Type 1     I_TH(1) = 1.02 / 1.61 = 0.63
Task Type 2     I_TH(2) = 0.41 / 0.85 = 0.48
Task Type 3     I_TH(3) = 0.61 / 0.81 = 0.75

Mean response time efficiency:

Task Type 1     I_ME(1) = 0.81 / 2.53 = 0.32
Task Type 2     I_ME(2) = 2.52 / 5.02 = 0.50
Task Type 3     I_ME(3) = 2.03 / 9.01 = 0.22

Timeliness efficiency:

Task Type 1     I_TI(1) = 0.50 / 1.61 = 0.31
Task Type 2     I_TI(2) = 0.35 / 0.85 = 0.41
Task Type 3     I_TI(3) = 0.60 / 0.81 = 0.75
```

Fig. 10-5 Efficiency values of the application software APS1

The efficiency values are significantly below 1. The values show that the application software APS1 is much less efficient than the reference software. The throughput is up to 52% lower (task type 2). The mean execution times are up to 78% longer (task type 3). The timely throughput is up to 69% lower (task type 1).

An important question is, whether the workload of the test bed was chosen to be too large with respect to the hardware speed. The answer is, that it was not too large because IP_0 fulfilled all user requirements. For brevity the $3*m = 3*3 = 9$ performance rating values (R values, see Section 7.7.1) of IP_0 are not printed here, but all are not less than 1.

The workload is appropriate. With another workload the test bed might have been inappropriate.

Here is a useful tip. From Fig. 10-3 it may seen at a glance that for each task types the E_0 value is not less than the B_0 value. I.e. all timeliness rating values (see equation (7-12) in Section 7.6) are not less 1. As this is so, it is a good omen for the system being satisfactory in fulfilling also the remaining user requirements. Applying this simple procedure to Fig. 10-4 we see, also at a glance, that the 30 users are not timely served when using the application software APS1. For each task type the E_0 value is less than the B_0 value.

10.4.1.3 The N_{max} oriented software efficiency value

This section deals with the efficiency value according to Section 10.3.3. The test bed and the measured systems IP_0 and IP_1 are the same as in Section 10.4.1.2 . But instead of measuring P_0 and P_1 we have to measure the maximum number of timely served users.

Firstly we install the reference software APS0 on the test bed. We have to determine the basic workload (see Section 8.2). Then a measurement series is performed by increasing stepwise the total number of users and N_{max0} determined. In our example N_{max0} equals 48.

The application SW APS0 is replaced by APS1. A measurement series is performed determining N_{max1} . In our example N_{max0} equals 24 .

The N_{max} related SW efficiency (according to equation (10.8) in Section 10.3.3) is shown in Fig. 10-6.

$$I_{maxuser} = 24 / 48 = 0.50$$

Fig. 10-6 N_{max} related software efficiency value of the application software APS1

This example shows that the actual application software APS1 is significantly less efficient than the reference SW APS0. If using APS1 the IP system can only serve 50% of the users timely (compare with APS0).

The two measured N_{max} values confirm the "tip" given at the end of Section 10.4.1.2 . The 30 users in that workload are timely served when using APS0 (because $N_{max0} = 48$ is greater than 30). The 30 users cannot be served timely when using APS1 (because $N_{max1} = 24$ is less than 30).

10.4.2 Example 2: System software efficiency

This example compares two sets of system software named SS-nU and SS-U. SS-U contains a UNIX based operating system with its system utilities and compilers etc. SS-nU contains a non UNIX based operating system also with its system utilities and compilers etc. The workload is the ISO workload "COMPCENTER1, Version M" as defined in ISO/IEC 14756 (see Section 11.3.2 of this book). This workload can be used for different operating systems. It was intentionally written for easy migration between different

operating systems. SS-U is the system software under test. SS-nU is the reference system software.

10.4.2.1 The measurement environment

The test bed consists of the components shown in Fig. 10-7 .

- user entity (example: 15 users)
- application SW
- place-keeper of
  a) Compilers and system utilities
  and b) Operating system
- HW

Fig. 10-7 Test bed for system software efficiency measurement

The user entirety in our example consists of 15 users. The workload has 5 task types. For details see Section 11.5.1 .

Installing the system software SS-nU onto the test bed produces the reference IP system IP_0. Changing SS-nU to SS-U produces the IP system IP_1.

10.4.2.2 The task-oriented software efficiency values

This section deals with the efficiency values according to Section 10.3.2.

An ISO-type performance measurement using IP_0 is performed. The measured performance is $P_0 = (B_0, T_{ME0}, E_0)$. Its values are shown in Fig. 10-8 .

Task Type	$B_0(j)$ tasks/sec	$T_{ME0}(j)$ sec	$E_0(j)$ tasks/sec
1	1.1024	1.74	0.1024
2	0.0512	1.80	0.0512
3	0.0512	4,67	0.0512
4	0.1024	1.48	0.1024
5	0.1024	1.30	0.1024

Fig. 10-8 The performance of the reference system IP_0
(i.e. using the reference system software)

Then the reference system software SS-nU is changed to the actual system software SS-U. The application software (written in higher languages) and the hardware all remain the same. This produces the system IP_1. An ISO-type performance measurement using IP_1 is performed. The measured performance is $P_1 = (B_1, T_{ME1}, E_1)$. Its values are shown in Fig. 10-9 .

Task Type	$B_1(j)$ tasks/sec	$T_{ME1}(j)$ sec	$E_1(j)$ tasks/sec
1	0.1100	0.72	0.1100
2	0.0550	1.35	0.0550
3	0.0550	1.95	0.0550
4	0.1100	0.86	0.1100
5	0.1100	0.80	0.1100

Fig. 10-9 The performance of the system IP_1
(i.e. using the actual system software)

The software efficiency values of SS-U result from the pair of measured performance values P_0 and P_1 . The computed values (see equations (10.5) to (10.7) in Section 10.3.2) are as shown in Fig. 10-10 .

Throughput efficiency

Task Type 1 $I_{TH}(1) = 0.1100 / 0.1024 = 1.074$
Task Type 2 $I_{TH}(2) = 0.0550 / 0.0512 = 1.074$
Task Type 3 $I_{TH}(3) = 0.0550 / 0.0512 = 1.074$
Task Type 4 $I_{TH}(4) = 0.1100 / 0.1024 = 1.074$
Task Type 4 $I_{TH}(5) = 0.1100 / 0.1024 = 1.074$

Mean response time efficiency

Task Type 1 $I_{ME}(1) = 1.74 / 0.72 = 2.4167$
Task Type 2 $I_{ME}(2) = 1.80 / 1.35 = 1.3333$
Task Type 3 $I_{ME}(3) = 4.67 / 1.95 = 2.3949$
Task Type 4 $I_{ME}(4) = 1.48 / 0.86 = 1.7209$
Task Type 5 $I_{ME}(5) = 1.30 / 0.80 = 1.6250$

Timeliness efficiency

Task Type 1 $I_{TI}(1) = 0.1100 / 0.1024 = 1.074$
Task Type 2 $I_{TI}(2) = 0.0550 / 0.0512 = 1.074$
Task Type 3 $I_{TI}(3) = 0.0550 / 0.0512 = 1.074$
Task Type 4 $I_{TI}(4) = 0.1100 / 0.1024 = 1.074$
Task Type 5 $I_{TI}(5) = 0.1100 / 0.1024 = 1.074$

Fig. 10-10 Software efficiency values of the system SW SS-U

All efficiency values are greater than 1. This shows that SS-U is more efficient than SS-nU. The throughput is 7.4% % better (for all task types). The mean response times are from 33% (task type 2) to 141% (task type 1) better. The timely throughput is 7.4% (for all task types) greater.

As in the example in Section 10.4.1, the question is whether the workload of the test bed was chosen to be too large with respect to the hardware speed. The answer is that it was not too large because IP_0 fulfilled all user requirements. For brevity the `3*m` = 3*3 = 9 performance rating values (`R` values, see Section 7.7.1) of IP_0 are not printed here, but all are not less than 1.

Compared with the N_{max} values the 15 users are low for both IP_0 and IP_1 (N_{max0} equals 18 and N_{max1} equals 45, see Section 10.4.2.3). Neither system is too heavily loaded. The weak load is the reason for the following fact: although the response times of IP_1 are much more better as those of IP_0 , the throughput of IP_1 is only 7.4% better. If the test bed had significantly more than 15 users the situation would be certainly very different.

Note 1: From Fig. 10-10 it can be seen that I_{TH} values of all task types are the same. This is no accident. It is a proper effect of the ISO workload "COMPCENTER1, Version M". Independent of the SUT to be measured the B values always have the ratio 2:1:1:2:2 . This is due to the definition of the chain probabilities and to the fact that there is only one user type. ■

Note 2: From Fig. 10-10 it can be seen that all I_{TI} values are also the same. This is due to the same reason above and because of the following. The 15 users are lower than N_{max} for both IP_0 and IP_1. All users are served timely and `B(j)` equals `E(j)` for each `j`. ■

10.4.2.3 The N_{max} oriented SW efficiency value

This section deals with the efficiency value according to Section 10.3.3. The test bed and the measured systems IP_0 and IP_1 are the same as in Section 10.4.2.2. But instead of measuring P_0 and P_1 we have to measure the maximum number of timely served users.

Firstly we install the reference system software SS-nU on the test bed. We have to determine the basic workload of "COMPCENTER1, Version M" (see Section 8.2). Then a measurement series is performed by increasing stepwise the total number of users, and N_{max0} determined. In our example N_{max0} equals 18.

The system software SS-nU is replaced by SS-U. A measurement series is performed determining N_{max1} . In our example N_{max0} equals 45.

The N_{max} related software efficiency is (according to equation (10.8) in Section 10.3.3) shown in Fig. 10-11.

$$I_{maxuser} = 45 / 18 = 2.50$$

Fig. 10-11 N_{max} related software efficiency value of the system software SS-U

This example shows that the actual system software SS-U is significantly more efficient than the reference software SS-nU. If using SS-nU the IP system serves 150% additional users timely (compare with SS-nU).

The two measured N_{max} values confirm the "tip" given at the end of Section 10.4.1.2 . The 15 users used in that workload are served timely when using both SS-nU (because 15 < N_{max0}) and SS-U (because 15 < N_{max1}).

10.5 Exercises

Exercise 1: Application software efficiency measurement using task-oriented software efficiency values. Example (1)

The application software:

- There are 3 activity types. They have the following functions. A defined text file is copied to the output file of the task. Additionally a search will be performed for all lines of the text file containing the letter "a" (activity type 1) or "b" (activity type 2) or "c" (activity type 3). The search result will be written in the output file.
- This application software is implemented by the set of the 3 shells TT1a, TT2a and TT3a of Exercise 1 in Chapter 8, Section 8.6 . Two different implementations can be simulated by use of different REP values. Applying a small REP value yields short run-times of the tasks. Applying a large REP value yields long run-times of the tasks. The basic function of this application software is always the same: copy the text file, search and write the result in the output file.
- Therefore this application software can simulate different implementations, depending on the REP value, but always having the same functionality.

Part 1
Use the user entirety of Exercise 1 in Chapter 8, Section 8.6 with N_{tot} = 10 users. The computer to be used is the same as in that exercise. The input data of the application software is also the same. For the operating system use LINUX. Define the test bed according to Figure 10.2 .

Part 2
Use REP=30 for the representation of the reference application software. This means installing TT1a, TT2a and TT3a and stipulating that in the reference measurement REP equals 30. This yields the reference system IP_0. Perform the reference measurement producing P_0 .

Part 3
Use REP=50 for representing the actual software to be measured. Replacing the reference application software by the actual software merely means stipulating that REP equals 50 in the measurement. Perform the measurement producing P_1.

Part 4:
Compute the task-oriented run-time efficiency values.

Exercise 2: Application software efficiency measurement using
task-oriented software efficiency values.
Example (2): changed reference application software
and changed actual application software.

The application software: Same as in Exercise 1.

Part 1
The task is the same as Part 1 of Exercise 1, but N_{tot} = 6 users.

Part 2
Use REP=50 for the representation of the reference application software. This yields the reference system IP_0. Perform the reference measurement producing P_0.

Part 3
Use REP=30 for the representation of the actual software to be measured. Perform the measurement producing P_1.

Part 4
Compute the task-oriented run-time efficiency values.

Exercise 3: Application software efficiency measurement using
N_{max} oriented software efficiency values,
Example (1).

Use the scenario of Exercise 1. Determine the value of the N_{max} oriented runtime efficiency term $I_{maxuser}$.

Exercise 4: Application software efficiency measurement using
the N_{max} oriented software efficiency values.
Example (2): changed reference application software
and changed actual application software.

Use the scenario of Exercise 2. Determine the value of the N_{max} oriented runtime efficiency term $I_{maxuser}$.

Exercise 5: Operating system efficiency measurement (using
N_{max} oriented efficiency values)

Note: Before solving this exercise read Chapter 11 and solve Exercises 1 and 2 in Section 11.6 .

Preparation:
Choose for the SUT a computer on which both a UNIX-type and a non-UNIX type operating system are available. The UNIX-type operating system is called OpSU and the non-UNIX system OpSnU. Make available an ISO-type RTE which can perform an ISO-type measurement with the OpSU-based SUT as well as with the OPSnU-based SUT.

Part 1
Use the ISO workload CC1. If necessary perform an OpSU migration in consideration of the migration rules (see Sections 11.4, 11.5 and 12.11).

Part 2
Migrate the ISO workload CC1 for OpSnU in consideration of the migration rules (see Sections 11.4, 11.5 and 12.11).

Part 3
Define the test bed according Figure 10-7 .

Part 4
Perform measurement series with each of the two systems and determine the N_{max} values for each of the workload versions `M, I, B` .

Part 5:
Taking the OpSnU for the reference system compute the N_{max} oriented software efficiency values $I_{maxuser}$ of the OpSU .

Solutions
For solutions see file
`CD/Solutions/Solutions-Section10-5.pdf .`

11 The ISO workloads

Annex F of ISO/IEC 14756 contains six ISO-type workloads. They are complete and ready for use for performing measurements. They also can be used for learning and understanding the ISO workload structure. The examples free the beginner from having to construct a workload before practising the ISO method. The 6 ISO workloads are (with permission of ISO) contained in the CD-ROM which is part of this book. They are explained in detail in Sections 11.2 and 11.3.

11.1 Purpose of the workloads and format

The format is briefly explained here. The table of contents of the workload is defined in Annex C of ISO/IEC 14756 which is a normative part of the standard. All workloads have to be written in this format. There are 6 sections:

1. Workload parameter set (WPS)
2. Application programs
3. Operating system command procedures
4. Computational results
5. Stored data
6. Statistical and accuracy parameters,
   called "advanced parameters" in Section 2.9 of this book

The machine readable representation of an ISO workload is a UNIX directory. It has six subdirectories. Additionally there is an introductory file and a subdirectory which contains tips for implementation. Therefore a workload directory contains one file and 7 subdirectories.

The workload directory structure is as follows (where "`wln`" represents the abbreviated name of the workload directory).

`wln/General`
This file contains a short description and an overview of the workload.

`wln/C1_WPS/`
This directory contains the WPS (workload parameter set) in a file also named WPS. There are 8 parts as follows.

- Basic parameters (see Section 2.6)
- Activity types (see Sections 2.3 and 2.7.1)
- Task types (see Sections 2.3 and 2.7.2)
- Timeliness functions (see Sections 2.5 and 2.7.3)
- chain types (see Sections 2.2, 2.4 and 2.7.4)
- Relative chain frequencies matrix (see Section 2.7.5)
- Preparation time mean values matrix (see Section 2.7.6)
- Preparation time standard deviations matrix (see Section 2.7.7)

The WPS is typically a text file having 8 sections. An example is shown in Fig. 11-1 . If the actual used RTE cannot read this text file it has to be rewritten manually in the format which the RTE can read.

Workload parameter set

1. Basic parameter values

(1) Total number of different user types: n = 3 ;
(2) Total amount of emulated users of each type:
N_user(1) = 25 ; N_user(2) = 40 ; N_user(3) = 20 ;
(3) Total number of different activity types: w = 4 ;
(4) Total number of different timeliness functions: p = 3 ;
(5) Total number of different task types: m = 5 ;
(6) Total number of different chain types: u = 4 ;

2. Activity type definitions

(1)Activity type number:	1	2	3	4
(2) The logical meaning of the input:	*)	**)	**)	**)
(3) The length (number) of characters) of the input string:	5	8	6	4
(4) The input string itself:	shb11	search q	NzzzAB	A5-9
(5) Activity type input variation:	none	none	yes ***)	none

*) Name of the shell script which has to be run.
**) Input string to be entered to the program.
***) zzz is a string of three random chosen decimal digits.

3.Task type definitions

(1) Current number j of the task type:	1	2	3	4	5
(2) Number of the activity type:	1	2	4	4	3
(3) Value of the task mode M(j): :	1	1	1	0	0
(4) Type number of the timeliness function:	1	2	2	3	1

4.Definitions of the timeliness functions

(1) Order number of the timeliness function:		1	2	3
(2) Number of time classes of this function:		z=2	z=3	z=2
(3) z couples of values g_t and r_t, where g_t is the time limit and r_t is the maximum accepted relative frequency:	g_t(1):	25 sec	50 sec	2.0 sec
	r_t(1):	0.90	0.90	0.90
	g_t(2):	62 sec	100 sec	4.0 sec
	r_t(2):	1.00	0.95	1.00
	g_t(3):		250 sec	
	r_t(3):		1.00	

Fig. 11-1 Example of a WPS

Workload parameter set (cont.)

5.Definitions of chain types

(1) The current number *l* of the chain type:	1	2	3	4
(2) The length L_chain(*l*) of the chain:	3	1	2	4
(3) The sequence of the task type numbers:	1	2	3	4
	3		3	3
	1			2
				2

6.Definition of the chain probabilities q(i,*l*)

l	q(1,*l*)	q(2,*l*)	q(3,*l*)
1	0.25	0.25	0.05
2	0.125	0.00	0.35
3	0.125	0.00	0.60
4	0.50	0.75	0.00

i = number of user type
l = number of chain type

7.Preparation time mean values h(i,j)

j	h(1,j)	h(2,j)	h(3,j)
1	25.0 sec	5.5 sec	17.0 sec
2	36.0 sec	20.5 sec	10.0 sec
3	40.0 sec	60.0 sec	5.0 sec
4	40.0 sec	3.0 sec	2.0 sec
5	40.0 sec	5.5 sec	90.0 sec

i = number of user type
j = number of task type

8.Preparation time standard deviations s(i,j)

	s(1,j)	s(2,j)	s(3,j)
1	5.0 sec	1.5 sec	6.0 sec
2	10.0 sec	5.0 sec	2.5 sec
3	10.0 sec	0.0 sec	0.5 sec
4	12.0 sec	0.0 sec	0.0 sec
5	12.0 sec	0.5 sec	1.0 sec

i = number of user type
j = number of task type

Fig. 11-1 (cont.) Example of a WPS

wln/C2_AppPrgs/
This directory contains all application programs used by the emulated users (see Section 2.8). The programs are ready for use on the SUT (either as an executable or the complete source code).

wln/C3_OSCP/
This directory contains the operating system command procedures (see Section 2.8). They are listed in full text, ready for use.

wln/C4_RES/
This directory specifies the correct values of all computational results (see Sections 2.8 and 2.9.1). There are two classes of activity types, depending on their input and computational results.

- Results of activity types having 'fixed input – fixed output':
  The input is the same whenever a task of this type is submitted and also the computational result. The result of the computation, in the case of correct operation of the SUT according to the input has to be listed for all these task types.
- Results of activity types having 'input variation':
  All variations of input or all rules have to be listened. Either all modified results of variations of input have to be listed, or all rules about how the output depends on the input have to be specified comprehensively and uniquely. So all correct results have to be described.

wln/C5_SData/
All data needed by the application programs, that are not contained in the input strings of the task types, are presented here (see Section 2.8). They are presented completely and in the final form ready to use on the SUT.

wln/C6_SPAP/
This directory contains the criteria for validation of a measurement. There are two classes.

- Subdirectory 'precision' contains the criteria for RTE accuracy (see Section 2.9.2). These are the DELTA values:
  x) $DELTA_q$, for the relative chain frequencies
  x) $DELTA_h$, for the preparation time mean values
  x) $DELTA_s$, for the preparation time standard deviations
- Subdirectory 'StatPar' contains the criteria for the statistical significance of the measurement (see Section 2.9.3). These are:
  x) the confidence coefficient ALPHA
  x) the m confidence intervals 2*d(j) of the mean execution times of the m task types (or d_{rel}, see Section 2.9.3).

wln/Prep/
This directory (in simple situations this might be a file) contains implementation tips or directions of how to make the application software on the SUT ready for a measurement run; for example how to use a data generator which converts the files of correct computational results into the specific data format of the actual SUT.

Final remark concerning the ISO workload structure:
The ISO/IEC 14756 is supplied on a CD. The six ISO workloads named SIMPLOAD1, SIMPLOAD2, SIMPLOAD3, COMPCENTER1, COMPCENTER2, COMPCENTER3 and

described in Annex F, are contained in directories named SL1, SL2, SL3, CC1, CC2, CC3 on the CD. These workload directories were created using a UNIX-SVR4 operating system. Important: Do not open or use these directories if using of another operating system, otherwise the data may be corrupted, the computational results in directory C4_RES can be distorted, programs may not run or produce incorrect results. When migrating a workload to a SUT having a non-UNIX operating system the workload directory has to be read by a UNIX system and converted for the new operating system (see Sections 11.4 and 11.5).

11.2 The Simple Workloads

11.2.1 General

Included in Annex F of ISO/IEC 14756 are three workloads called SIMPLELOAD1, SIMPLOAD2, SIMPLOAD3. These workloads are primarily intended as examples for showing the structure of ISO-type workloads. They are not really intended for measuring and rating the performance of current computers.

11.2.2 SIMPLOAD1

This workload is found in the directory SL1/ in Annex F. It has a complex WPS but uses very simple activity types. This workload is intended for

- gaining a better understanding of the ISO measuring method
- testing the functional correctness of an RTE

Although this workload is written for SUTs with a UNIX-SVR4 type operating system, it can be easily migrated to any other multi-user operating system. The application program is written in C, but it can be easily converted to another programming language.

11.2.3 SIMPLOAD2

This workload is found in the directory SL2/ in Annex F. It uses very simple activity types. This workload is intended for

- gaining a better understanding of the ISO measuring method
- testing the functional correctness of an RTE

This workload is very similar to SIMPLOAD1 but it has only one user type. Therefore the number N_{tot} of emulated users can be modified by steps of 1. This is contrary to SIMPLOAD1, the number N_{tot} of emulated users can be modified by steps of 7 . SIMPLOAD2 may be used to show the effect of varying the number of users by steps of 1.

Although this workload is written for SUTs with a UNIX-SVR4 type operating system, it can be easily migrated to any other multi-user operating system. The application program is written in C, but it can be easily converted to another programming language.

11.2.4 SIMPLOAD3

This workload is found in the directory `SL3/` in Annex F. It only differs from SIMPLOAD2 in having less efficient programs, but the same functionality. The effect of less run-time efficiency of the application software may be shown by comparing the measurement and rating results with those of the workload SIMPLOAD2 (See Chapter 10, Measurement of software run-time efficiency).

Although this workload is written for SUTs with a UNIX-SVR4 type operating system, it can be easily migrated to any other multi-user operating system. The application program is written in C, but it can be easily converted to another programming language.

11.3 The Computer Centre Workloads

11.3.1 General

Included in Annex F of ISO/IEC 14756 there are three workloads called COMPCENTER1, COMPCENTER1 and COMPCENTER3. They are found in the directories `CC1/`, `CC2/` and CC3/ . Contrary to the SIMPLOADs they are primarily intended for performance measurement as well as for demonstration. These workloads use flat files in order to be independent from any data base system. Although the application software seems to be simple, these workloads have proved suitable for the performance measurement of computers of all sizes. This is possible by adjusting the CPU loading with a control variable named `REP` (replication factor).

11.3.2 COMPCENTER1

This workload uses COBOL and FORTRAN application programs for activity types. It represents a job stream profile typical for computer centre operation with program development and heavy production work. The workload has 5 task types. The CPU loading of the workload can be adjusted by a control variable called `REP` .

This workload has only one user type. The advantage of this is that the number N_{tot} of users can be modified by steps of 1. This workload can also be used to show the effect of varying the number of emulated users by steps of 1. By performing a series of measurements with increasing numbers of users it is possible to determine the maximum number of timely served users N_{max} with a precision of 1. This workload is written for SUTs using a UNIX-SVR4 type operating system. But it can be easily migrated to any other multi-user operating system. For more details see Section 11.5.1.

There are three versions of this workload, "Version M", "Version I" and "Version B" (see Section 11.5.1).

11.3.3 COMPCENTER2

The characteristics of this workload are similar to those of COMPCENTER1, but the application programs are written in C, instead of COBOL and FORTRAN. In contrast to COMPCENTER1, a special design of the WPS demonstrates how to realise batch mode for defined task types. The CPU loading of the workload can be adjusted by a control variable called REP . This workload is written for SUTs using a UNIX-SVR4 type operating

system. But it can be easily migrated to any other multi-user operating system. For more details see Section 11.5.2.

11.3.4 COMPCENTER3

The application program of this workload is a simple OLTP system (OLTP online transaction processing). The workload represents a user entirety which is typical for OLTP computer centre operation. For reasons of simplicity there is no use of a database system. The application uses "flat files".

This workload has 2 user types. There are 4 times as many type 2 users as type 1 users, or

$$N_{user}\ (2)\ /\ N_{user}\ (1)\ =\ 4\ .$$

This workload can also show the effect of varying the number of users. Multiplying both $N_{user}\ (1)$ and $N_{user}\ (2)$ by factors of 2, 3, 4, ... , the total number of users increases in increments of 5. By performing a series of measurements using increasing numbers of users the maximum number N_{max} of timely served users can be estimated. In most cases there is only a small disadvantage when the number of users can only be varied by steps of 5. This is due to the fact that a SUT typically serves a great number of users of this workload. Therefore the relative failure of the estimated N_{max} arising from the increase by steps of 5 is not too dramatic. The CPU loading of the workload can be adjusted by a control variable called `REP`. This workload is written for SUTs using a UNIX SVR4 type operating system. But it can be easily migrated to any other multi-user operating system. The application software is available in both COBOL and C. For more details see Section 11.5.3.

11.4 Migration of ISO workloads to other operating systems

The ISO workloads are written for SUTs using a UNIX-SVR4 operating system. They can be easily migrated to other UNIX type operating systems, and also to any multi-user operating system.

"Migration of an ISO-type workload" does not mean converting it somehow to a version which runs on the new operating system. It means creating an implementation which ensures that all steps are converted without changing the logical structure. A step shall neither be changed nor omitted. Steps shall not be merged to a composite step.

The migration rules are:

1. The directory structure of the workload must not be changed in any detail.
2. The contents of the file '`wln/General`' have to be adapted to the new operating system. The name of the person and the organisation responsible for the migration have to be cited together with the date of the migration. The original version has to be cited. A reference has to be given where the complete original version is available for public access. This is very important if the source version for the migration was not an original ISO workload.
3. The contents of the directory '`wln/C1_WPS/`' have to remain unchanged in all details.
4. The contents of the application program directory '`wln/C2_AppPrgs/`' must not be changed. The programs must not be modified at all. No changes of the program logic or

the processing sequence are allowed. In case of executables need to be recompiled, the same compiler options as for the original executables must be used.

5. The contents of the operating system command procedures directory ‘`wln/C3_OSCP/`’ will most likely have to be changed because of the new command language. The command procedures will have to be converted provided that there is no change in the sequence of steps and that each step performs exactly the same function. Changes of symbolic names, such as file names and environment variables, are allowed. But these must not influence the program logic or the command procedure sequence logic.
6. The contents of the result directory '`wln/C4_RES/`' must not be changed.
7. The contents of the stored data directory '`wln/C5_SData/`' must not be changed.
8. The contents of the accuracy criteria directory '`wln/C6_SPAP/`' must not be changed.
9. The contents of the preparation tips directory '`wln/Prep/`' usually has to be adapted to the new operating system. But only unavoidable changes may be made provided they do not change the process defined in the original workload.
10. If, in course of the migration, any deviation from the rules 1 to 9 has occurred, the following must be documented in file ‘`wln/General`’:
    a) Explicit declaration of why, where and how the workload was modified in the course of the migration.
    b) Detailed explanation of all changes.

11.5 Important details for ISO workload migration

Note: The following Sections 11.5.1 to 11.5.3 describe details of the workloads that are important for migration. Additionally they can be used as tips on how to migrate other ISO type workloads, for instance the ISO’s SIMPLOADs (see Section 11.2) as well as individual ISO-type workloads (see Chapter 12). ■

11.5.1 Workload COMPCENTER1

11.5.1.1 Introduction

The ISO workload COMPCENTER1 was originally written for the operating system UNIX-SVR4. The changes to be made when migrating to another operating system concern primarily the operating system command procedures (OSCPs). Few changes will be needed to the rest of the workload. For transferring the OSCPs into the new command language they have to be analysed for their logical steps. These steps have to be rewritten in the new command language (see Section 11.5.1.2).

The workload contains 5 activity types, represented by 5 OSCPs (ta's) as below. The input string (see Fig. 2-9 in Chapter 2) of each is the name of a command procedure.

- "`ta1`": This simulates a programmer editing his program source. As an example a text is chosen which looks like a FORTRAN program. The logical meaning of the text is unimportant, but the text must no be changed. The only goal is to execute defined text operations like text input, search, replace etc. using a defined text.
- "`ta2`": This simulates a programmer testing his COBOL program. He compiles it and then starts the generated executable. The program is intended to represent a business task..

- "ta3": This simulates a programmer testing his FORTRAN program. He compiles it and then starts the generated executable. The program is intended to represent a technical or scientific task.
- "ta4": This simulates a computer centre job (coming from a FORTRAN program). It executes a tested executable. The job is intended to represent a technical or scientific task.
- "ta5": This simulates a computer centre job (coming from a COBOL program). It executes a tested executable. The job is intended to represent a business task.

The WPS of this workload defines a very simple construction of its task types as follows. Activity type x is only used by task type x (x = 1, 2, ..., 5). Task types 1, 2 and 3 use M = 1. Task types 4 and 5 use M = 0 . M is the task mode. This construction tempts the reader to guess that generally "ta1", "ta2" and "ta3" are interactive jobs and "ta4" and "ta5" are batch jobs. But it is not so. Please remember that the task mode values M = 0 or 1 are not generally identical with the traditional modes "batch" or "interactive" (see Section 2.3 and compare to Section 11.5.2.1). Therefore whether a task is interactive or placed to the background depends on the task mode of the preceding task. This is the situation when the workload COMPCENTER1 is operated in its usual version called "Version M". Please do not confuse this name 'Version M' with the task mode variable M.

The workload COMPCENTER1 has two additional versions, "Version I" and "Version B". "Version I" activates only the task types 1, 2 and 3. This is achieved by setting the relative chain frequencies q(1,4) and q(1,5) to zero. With "Version I" all tasks submitted to the SUT use the task mode with the value M = 1. This causes all tasks to operate identically to the traditional mode "interactive". Contrary to "Version I" the "Version B" activates only the task types 4 and 5. This is achieved by setting the relative chain frequencies q(1,1), q(1,2) and q(1,3) to zero. With "Version B" all tasks submitted to the SUT use the task mode with the value M = 0. This causes all tasks to operate identically to the traditional mode "batch".

Studying these details will provide a better understanding of the workload and its OSCPs.

11.5.1.2 The logical steps of the OSCPs of COMPCENTER1

All logical steps of the 5 OSCPs are defined by its original text in Appendix F of ISO/IEC 14756, written in UNIX command language. They have to be migrated to the new operating system following exactly the migration rule 5 of Section 11.4 . For providing a better understanding the steps they are shortly described in the Appendix A of this book, see Section 1 in file

CD/Supplement-to-ISO14756.pdf .

For additional information concerning the preparation procedures and the installation of the workload COMPCENTER1 on the SUT see ISO original text in directory

CD/iso14756-orig-workloads/ .

11.5.1.3 Some explanations

11.5.1.3.1 Preparation times (think times) and task modes of the 5 task types

Each task type consists of a sequence of steps. To facilitate using the RTE and the workload, the preparation times are somewhat simplified and treated as follows. There are no preparation times between the steps. Instead the sum of them is represented by the preparation time before the submission of each task. Consequently, the preparation times refer to each of the task types as one complete action and not to its separated steps (see Special case 1 in Section 2.7.1). The mean values of the 5 task types and their corresponding standard deviation values are found in the WPS (in the directory `CC1/C1_WPS/`).

11.5.1.3.2 Execution times (response times) of the five task types

The execution times are treated analogously to the preparation times. No separate execution times of each step of a task type are measured. Instead an execution time is defined which refers to the complete action of the task type and not to its separated steps (see Special case 1 in Section 2.7.1).

11.5.1.3.3 The function "<process#>"

The penultimate step of OSCP is to copy the computational result into a file of the user home directory. These files have the suffix `<process#>`. This suffix is the actual UNIX process number of the SUT. The suffix ensures that all stored files have different names, even if there is a large number of users and each user executes a task type many times. When migrating a workload to a non-UNIX operating system it is important to check if this system supports the function of accessing to system wide unique process numbers. If not, a new solution has to be developed which ensures that all computational results are stored and none of these files will be overwritten.

11.5.1.3.4 Submission of the value of REP to the SUT

The replication factor "`REP`" (in the ISO UNIX version of the workload COMPCENTER1) is transferred from the RTE to the SUT by an UNIX environment variable. The implementation of this feature is somewhat dependent on the realisation of the RTE. If migrating the workload to a new operating system this solution will eventually not work. Then a new solution has to be developed. It is important that this solution does not change the CPU loading of the SUT compared with the generic ISO workload version.

11.5.1.3.5 Size of the tasks of COMPCENTER1

Typically, computer centre production jobs run for a long time, ranging typically from a quarter of an hour to several hours. If such tasks occur in a workload, the rating period is very long in order to include a sufficient number of such tasks (necessary for sufficient statistical significance of the measured performance values). In order to avoid extremely long rating periods this workload contains, instead of a few long production jobs, a large number of short production jobs. Experience has shown that a rating period length of significantly less than one hour is sufficient.

11.5.2 Workload COMPCENTER2

11.5.2.1 Introduction

The ISO workload COMPCENTER2 (summarised in Section 11.3.3) was originally written for the operating system UNIX-SVR4. This workload is similar to COMPCENTER1 but instead of FORTRAN and COBOL programs it uses programs written in the language C. The changes to be made when migrating to another operating system concern primarily the OSCPs. Few changes will be needed to the rest of the workload. For transferring the OSCPs into the new command language they have to be analysed for their logical steps. These steps have to be rewritten in the new command language (see Section 11.5.2.2).

The workload contains 6 activity types, represented by 6 OSCPs (ta's) as below. The input string (see Fig. 2-9 in Chapter 2) of each is the name of a command procedure.

- "`ta1`": (same as ta1 of COMPCENTER1) This simulates a programmer editing his program source. As an example a text is chosen which looks like a FORTRAN program. The logical meaning of the text is unimportant, but the text must not be changed. The only goal is to execute defined text operations like text input, search, replace etc. using a defined text.
- "`ta2`": This simulates a programmer testing his C program. He compiles it and then starts the generated executable. The program is intended to represent a business task.
- "`ta3`": This simulates a programmer testing another C program. He compiles it and then starts the generated executable. The program is intended to represent a technical or scientific task.
- "`ta4`": This simulates a computer centre job (coming from a C program). It executes a tested executable. The job is intended to represent a technical or scientific task.
- "`ta5`": This simulates a computer centre job (coming from a C program). It executes a tested executable. The job is intended to represent a commercial task.
- "`ta0`": This is a dummy procedure for special use (see the following explanation).

Similar to COMPCENTER1 the WPS of this workload defines a very simple construction of its task types as follows. Activity type `x` is only used by task type `x` (`x` = 1, 2, ..., 5). All these task types use the task mode with the value `M` = 1. This construction tempts the reader to guess that generally these task types are interactive jobs. But this is not so. There is an additional task type 6 which has an empty activity type. The task mode has the value is `M`=0. The task types 4 and 5 are always followed by task type 6 (see the chain definitions of the WPS). This results in "`ta4`" and "`ta5`" always going into the background. I.e. they are executed as a traditional batch job. (For additional explanations see Section 2.3 and compare to Section 11.5.1.1 and to Appendix A, Section 2.6 in file `CD/Supplement-to ISO14756.pdf`). This is the situation when the workload COMPCENTER2 is operated in its usual version called "Version M". Please do not confuse this name "Version M" with the task mode variable `M`.

The workload COMPCENTER2 has two additional versions, "Version I" and "Version B". "Version I" activates only the task types 1, 2 and 3. This is achieved by setting the relative chain frequencies `q(1,4)` and `q(1,5)` to zero. With "Version I" all tasks submitted to the SUT use the task mode with the value `M` = 1. This causes all tasks to operate identically to the traditional mode "interactive". Contrary to "Version I" the "Version B" activates only the task types 4 and 5. This is achieved by setting the relative chain frequencies

`q(1,1)`, `q(1,2)` and `q(1,3)` to zero. With "version B" all tasks submitted to the SUT are put into the background. This causes all tasks to operate identically to the traditional mode "batch".

Studying these details will provide a better understanding of the workload and its OSCPs.

11.5.2.2 The logical steps of the OSCPs of COMPCENTER2

All logical steps of the 6 OSCPs are defined by its original text in Appendix F of ISO/IEC 14756, written in UNIX command language. They have to be migrated to the new operating system following exactly the migration rule 5 of Section 11.4 . For providing a better understanding the steps they are shortly described in the Appendix A of this book, see Section 2 in file

`CD/Supplement-to-ISO14756.pdf` .

For additional information concerning the preparation procedures and the installation of the workload COMPCENTER2 see ISO original text in directory

`CD/iso14756-orig-workloads/` .

11.5.2.3 Some explanations

11.5.2.3.1 Preparation times (think times) and task modes of the 6 task types

To facilitate using the RTE and to simplify the workload the preparation times are also simplified handled analogously to the workload COMPCENTER1 (see Section 11.5.1.3.1).

11.5.2.3.2 Execution times (response times) of the 6 task types

Analogously to the preparation times, the execution times are simplified as in workload COMPCENTER1 (see Section 11.5.1.3.2).

11.5.2.3.3 The function "<process#>"

This is the same procedure as in workload COMPCENTER1 (see Section 11.5.1.3.3).

11.5.2.3.4 Submission of the value of REP to the SUT

The replication factor "`REP`" (in the ISO UNIX version of the workload COMPCENTER1) is transferred from the RTE to the SUT by a UNIX environment variable. The idea is the same as in workload COMPCENTER1 (see Section 11.5.1.3.4). The implementation of this function is somewhat dependent on the realisation of the RTE. If migrating the workload to a new operating system this solution might not work and a new solution has to be developed. It is important that this solution does not change the CPU loading of the SUT compared with the generic ISO workload version.

11.5.2.3.5 Size of the tasks of COMPCENTER2

The same idea is used as in the workload COMPCENTER1 (see Section 11.5.1.3.5).

11.5.3 Workload COMPCENTER3

11.5.3.1 Introduction

The workload COMPCENTER3 was originally written for the operating system UNIX-SVR4 and is summarised in Section 11.3.4 . The main changes to be performed, when migrating to an other operating system, refer to a few OSCPs needed for preparing the workload on the SUT. The activity types do not use any command procedures. Therefore no changes are necessary. The application programs remain unchanged.

The application software of this workload is available in two languages, COBOL-ANS85 and ANSI-C. There are only two activity types: activity type 1 (read data from the data base) and activity type 2 (write data into the data base).

11.5.3.2 The logical steps of the OSCPs of COMPCENTER3

There are only two pairs of very simple OSCPs. One pair is for installing the COBOL version of the application software and the other one is for the C version. These procedures have to be rewritten for migration of the workload to other operating systems.

The COBOL related OSCPs
The procedure "`compile.gencob`" generates the object code "`generate`" from the COBOL source file "`generate.cob`". This is the data generation program. The procedure "`compile.taxcob`" generates the object code "dialog" from the COBOL source file "`dialog.cob`". This is the application program. The compiler options may be chosen freely.

The C related OSCPs
The procedure "`compile.genc`" generates the object code "`generate`" from the C source file "`generate.c`". This is the data generation program. The procedure "`compile.taxc`" generates the object code "dialog" from the C source file "`dialog.c`" . This is the application program. The compiler options may be chosen freely.

11.5.3.3 Installation of the workload COMPCENTER3 on the SUT

The installation steps are described in sufficient detail in the file CC3/Prep/readme of the original ISO workload description.

11.5.4 Migration examples of ISO workloads

ISO workloads were migrated to various UNIX and non-UNIX operating systems. These versions are typically owned by the companies which performed the migrations and are not

available. Some versions are available, see Appendix A of this book, directories

```
CD/Linux-workloads/
```

and `CD/NT-workloads/` .

For some drafts (incomplete sketches) see Solutions of Exercise 4 in Section 11.6. These may be helpful in performing an actual migration.

11.6 Exercises

Exercise 1: Implementation of an ISO workload on a UNIX system

Part 1
Migrate one or more (but at least the second) of the ISO workloads COMPCENTER1, COMPCENTER2, COMPCENTER3 to a UNIX operating system available to you. For migration rules see Sections 11.4 and 11.5 .

Part 2: test the migrated workloads.

Exercise 2: Measurement

Perform a measurement series using the migrated workload COMPCENTER2-M of Exercise 1. Use the measurement system DEMO. Determine the performance value N_{max} of your SUT.

Exercise 3: Migration to a midrange system

Part 1
Become familiar with the operating command language of a non-UNIX operating system of a midrange computer. Migrate the ISO workload COMPCENTER2 to this operating system. For migration rules see Sections 11.4 and 11.5 .

Part 2 : test the migrated workload.

Exercise 4: Migration of an ISO workload to a traditional mainframe system

Part 1: become familiar with the operating command language of a mainframe operating system. Migrate one or more of the ISO workloads COMPCENTER1, COMPCENTER2, COMPCENTER3 to this operating system. For migration rules see Sections 11.4 and 11.5 .

Part 2
Test the migrated workloads.

Solutions
For solutions see file

```
CD/Solutions/Solutions-Section11-6.pdf  .
```

12 Creating an individual ISO-type workload

Annex F of ISO/IEC 14756 consists of example workloads that can be used as standard workloads. There may, however, be situations where the user entirety is not or not sufficiently be represented by one of these reference workloads. Therefore an individual workload has to be defined.

It is assumed that before this Chapter 12 is read, the ISO user data model has been thoroughly understood by studying in particular Chapter 2 and Section 11.1.

12.1 Activity types and their representatives

Defining an individual ISO-type workload starts with listing all users of the regarded user entirety. A user can be a person (a human user) or a machine (data processing system). As the user entirety typically changes depending on the type and time of day, the time period has to be specified. For instance, this could be the peak traffic rush hour of the January weekday mornings from 11.00 to 12.00.

For this period each user has to be analysed for the activities submitted to the SUT. (Please remember Section 2.3; an activity is an order submitted to the SUT by a user for the execution of a defined data processing operation.) Perhaps this produces a large number of activities. The next step is to reduce this list considerably.

The various activities have to be classified into types. It is strongly recommended that as few as possible activity types are defined. In many cases 3 to 10 activity types are sufficient. Even with a large number of very different users you should not have more than 30 or 40 activity types. Experience has shown that more than 100 activity types are a disadvantage for the purpose of performance measurement and rating. For each of the activity classes a typical representative has to be chosen. These representatives are the basis for defining the workload. The number of representatives is the number w of activity types in the WPS (see Section 2.6). This is the basic parameter (3) in part 1 of the file `wln/C1_WPS/wps` (see Section 11.1 and Fig. 11-1). For each of the w activity types there is a column of 5 entries in part 2 of the file. The fifth entry indicates whether the activity type has input variation. If there is input variation, the rule of it also has to be specified (see Section 2.7.1).

12.2 Timeliness functions

The next step of defining an ISO-type workload is to specify the timeliness functions. For each of the users, and for each of the activity types used by him, a timeliness function (see Section 2.5) has to be defined. Note that a user may use all activity types. Perhaps the analysis would produce a large number of timeliness functions. But this number can be substantially reduced by identifying and selecting typical timeliness functions. Usually only a few are needed. For instance: "fast response", "medium response", "slow response", "background job response", "priority job response". The execution time requirements of most users are described sufficiently by using these typical timeliness functions. It is recommended that 2 or 3 time classes, but not only one and not more than 4, are used (see Section 2.5).

The chosen number of timeliness functions yields the value of p. It is the basic parameter (4) in part 1 of file `wln/C1_WPS/wps` (see Section 11.1 and Fig. 11-1). From the investigation also result the p activity type definitions in part 4 of this file, which contains one column of entries for each timeliness function. Each column consists of the current number of the timeliness function, the number `z` and `2*z` values where z is the number of time classes of the timeliness function (see Section 2.7.3).

12.3 Task types

The next step in defining a workload is to determine the task types. For each user the following has to be defined: For each pair of activity type and timeliness function the value of the task mode M has to be determined. Note that not all pairs may be used by the user. Only for used pairs M has to be defined. For M see Section 2.3. It specifies whether the user starts his preparation time (which precedes the task submission) immediately after the submission of the previous task (`M=0`), or if he starts his preparation time when the previous task has been completed (`M=1`). For a discussion about task submission and task completion see Section 3.4 .

For a comparison of `M=0` or `M=1` with batch or interactive jobs see Sections 2.3 and 11.5.1.1. See also Section 11.5.2.1 for the device of the of the dummy activity type 6.

Each of the triples "activity type, timeliness function, task mode" represents a task type (see Section 2.3). Perhaps this analysis would produce a large unmanageable number of task types. But this number can be substantially reduced by identifying and selecting typical task types.

It is strongly recommended that as few as possible task types are defined. In many cases 3 to 10 activity types are sufficient. Even with a large number of very different users you should not have more than 30 or 40 task types. Experience has shown that more than 100 task types are a disadvantage for performance measurement and rating.

The chosen number of task types yields the value of m. It is the basic parameter (5) in part 1 of file `wln/C1_WPS/wps` (see Section 11.1 and Fig. 11-1). From the investigation also result the `m` task type definitions in part 3 of this file, which contains one column of entries for each timeliness function. Each column consists of the 4 entries task type number, activity type, task mode and timeliness function.

12.4 Chain types

So far the analysis has reduced the real user entirety to an abstract user entirety which uses only the defined m task types. The ISO workload model assumes that the users typically submit sequences of tasks (see Section 2.4). Such sequences are task chains. Therefore we have to analyse all users as defined in the reduced user entirety, with respect to the sequences they submit. This again seems to produce a large number of different sequences. But this number can be reduced by identifying and selecting typical chains. Select only the most important ones. Usually a moderate number is sufficient. As a rule do not define more than 10 chain types to be used by a user. This rule follows from the necessity of having not too small chain probabilities (see Section 12.5). Elsewhere the rating interval,

i.e. the duration of the measurement, would be very long. This analysis yields the value of u, the total number of chain types.

u is the basic parameter (6) in part 1 of file wln/C1_WPS/wps (see Section 11.1 and Fig. 11-1). From the analysis also result the u chain type definitions in part 5 of this file, which contains one column of entries for each chain type. Each column consists of the 3 entries chain type number, length of the chain (number of tasks) and the task type sequence itself.

12.5 Chain probabilities and user types

Each user has to be considered with respect to his relative frequencies (q values) of using chain types. This yields an initial column of u entries for each user. Each of the u entries is a value between zero and 1 and the sum of all values has to be 1. Comparing all these columns will show that many of them are similar or nearly identical. That means that, with respect to the q values, there are subgroups of the user entirety. The members of a subgroup use each chain type with (nearly) the same relative frequency. The subgroups are called user types (see Section 2.6). This analysis yields the total number of user types called n. Its value is the basic parameter (1) of part 1 of the file wln/C1_WPS/wps (see Section 11.1 and Fig. 11-1). The investigation yields also the total number of users of each type which are the n values of the basic parameter group (2).

Based on the similar q value columns (of all users of one type) a common column, valid for all users of the type, has to be defined. It is strongly recommended, for getting not too long rating intervals, to chose for the relative frequencies values which are not too small. Generally, values should be multiples of 0.05 .

The relative chain frequencies of the n user types found, appear in the n columns of part 6 of the file wln/C1_WPS/wps . Each column has u entries. The sum of the values of each column has to equal 1. This is the chain probability matrix ("p-matrix").

The reason why the q values should not be too small is made clear by the following examples. Let the number of chain types be u=3 .

Example 1: Let the q values of a defined user type be 0.04, 0.93 and 0.03 . The greatest common divisor of these three values defines the minimum number of chains in the RI for having a chance of realising exactly the desired relative chain frequencies. The greatest common divisor is 0.01 . Therefore

1.0/0.01 = 100 chains (or a multiple thereof)

must be included in the RI. The desired relative frequencies are realised if having 4, 93 and 3 chains of the types 1, 2 and 3 in the RI.

Example 2: Let the q values of the defined user type be 0.40, 0.20 and 0.60 . The greatest common divisor is 0.20 . Then at least

1.0/0.20 = 5 chains (or a multiple thereof)

must be included in the RI.

It is obvious that the second example, which has larger q values, yields a much shorter RI.

12.6 Preparation times ("think-times")

Each submission of a task is preceded by the user's preparation time (i.e. think time) as explained in Section 2.7.6 . Depending on both the user type and task type, the mean value of the preparation time has to be determined from the user entirety. This produces a list of m values for each user type. The list is represented as a column in the WPS. For `n` user types are there `n` columns. This is the preparation time mean values matrix (`h`-matrix) of part 7 of the file `wln/C1_WPS/wps` (see Section 11.1 and Fig. 11-1).

In addition to the mean values of the preparation times the standard deviation values have to be specified (as explained in Section 2.7.7). A standard deviation value has to be defined corresponding to each element of the preparation time matrix. This produces the n columns of m values of part 8 of the file `wln/C1_WPS/wps` which is the preparation time standard deviation matrix (`s`-matrix).

The standard deviation is an indicator of how much the randomly created preparation times differ from the mean value. A standard deviation value of zero means that all preparation times are equal to the mean value. The `s` values may not be too large in relation to their corresponding `h`-values (see Section 2.7.7). As a rule an `s` value should not be greater than 50% of its corresponding `h`-value. When using the Urn Method the limit is a little larger (as shown in Sect. 6.3.3).

12.7 The WPS

12.7.1 Recording the values of the WPS in a text file

The values of the WPS have to be recorded in ISO format in a text file named `wps` . These values are determined according to Sections 12.1 to 12.6 above, and summarised as follows.

Section 12.1 produces
- the value of `w`, basic parameter (3) in part 1 of the `wps` file
- the activity type definitions in part 2 of the `wps` file.

Section 12.2 produces
- the value of `p`, basic parameter (4) in part 1
- the timeliness functions in part 4.

Section 12.3 produces
- the value `m`, basic parameter (5) in part 1
- task type definitions in part 3.

Section 12.4. produces
- the value of `u`, basic parameter (6) in part 1
- chain type definitions in part 5.

Section 12.5 produces
- the value `n`, basic parameter (1) in part 1
- the user number of each type, the `n` values of basic parameter group (2) in part 1
- chain probabilities in part 6 (`q`-matrix).

Section 12.6 produces
- preparation time mean values in part 7 (h-matrix)
- preparation time standard deviations in part 8 (s-matrix).

An example of a WPS can be found in Fig. 11-1 .

12.7.2 Recursive improvement of the WPS

After defining this first version of the WPS at least a further attempt is recommended in order to simplify it. The goal is to reduce the values of

n (number of different user types),
w (number of different activity types),
p (number of different timeliness functions),
u (number of different chain types)

while preserving the essentials of the real user entity.

Another goal is to reduce the measurement duration. This depends among other things on the preparation times. Long preparation times result in long measurement durations. (And, naturally, the execution times of the SUT influence the measurement duration; but these times are the subject of the measurement and not of the WPS specification.) If there is a very large h value, then check if it would be possible to replace the corresponding task type by a number of shorter running tasks and reduce accordingly the corresponding mean preparation time. Then a greater number of execution time samples per hour are produced and there is a better chance of fulfilling the ISO criteria of statistical significance of the measurement within a shorter measurement duration.

The number of tasks per hour of a task type does not only depend from the preparation times and the execution times. If there are small values in the q-matrix, a long time is needed for obtaining a suitable number of tasks of the related task types. Therefore check the q-matrix for small values less than 0.05 (see Section 12.5). If there are any, see if the corresponding chain types can be replaced in order to obtain a larger q-value.

As also mentioned in Section 12.5 another rule is that the values of the q-matrix should be a multiple of 0.05. It would be even better if the q-matrix values were a multiple of at least 0.1 . If possible make this adjustment to the q-matrix.

This improves the chance for achieving "OK" with respect to the $DELTA_q$ criterion (see Section 4.2.2) with a shorter measurement duration especially when using the Urn Method (see Chapter 6).

For a roughly estimate of the magnitude of the measurement duration the following procedure can be used: Insert the WPS in your ISO RTE (for instance DEMO) and let it compute the throughput vector of the ISO reference machine (see Sections 7.2 and 7.3.2)

$$B_{Ref} = (B_{Ref}(1), B_{Ref}(2), \ldots, B_{Ref}(m)) \quad .$$

Search for the smallest of these m values. For instance the value could be

$$B_{Ref}(3) = 0.004 \text{ tasks/sec} \quad .$$

This means that the most infrequent task type is submitted about 15 times per hour. Usually about a dozen samples are needed for getting an acceptable statistical significance. Assume that the SUT is fast enough for fulfilling the timeliness requirements of the workload. Then the measured throughput approximate the magnitude of the reference throughput. The dozen samples of the most infrequent task type needed will be produced in less than an hour (RI). This would be an acceptable measurement duration. But, on the other hand, if the smallest reference throughput value is 0.0004 tasks/sec then the measurement duration could be 9 to 10 hours (RI) which might be too long. Then check if the WPS can be redesigned.

12.8 The application software

The application SW and the related data have to be documented in the workload directories

```
wln/C2_AppPrgr/
wln/C3_OSCP/
wln/C4_RES/
wln/C5_SData/    .
```

Follow the explanations of Sections 2.8 and 11.1 and use the workloads described in Sections 11.2 and 11.3 as examples.

12.9 The advanced parameters

These parameters check the sufficiently precise working of the RTE and the statistical significance of the measured results. The values have to be stored in the directory

```
wln/C6_SPAP   .
```

For the format see the examples of the Sections 11.2 and 11.3. For the values follow the explanations and recommendations in Section 2.9 .

12.10 Preparation

The content of the directory `wln/Prep/` can be designed individually according to the requirements of the workload (see Section 11.1). For examples see Sections 11.2 and 11.3 .

12.11 Migration of an individual ISO-type workload to a different operating system

An individual ISO-type workload is typically developed for and implemented on an operating system of primary interest to its designer. However, interest often arises for migrating this workload to a different operating system. For such an action the same rules are valid as explained with respect to the ISO workloads in Section 11.4. There is no difference between individual ISO type workloads and the workloads published in ISO/IEC 14756. Before migrating it is essential to analyse thoroughly the operating system command procedures. For examples see Sections 11.5.1, 11.5.2 and 11.5.3 and Annex A, see file

```
CD/Supplement-to-ISO14756.pdf     .
```

12.12 Exercises

Exercise 1: Specifying a scenario and defining the according ISO-type workload

Suppose a LINUX computer is used with a simple application of your own choice. Sketch roughly in your own words the user entirety: scenario. Create the ISO type workload and check it so far as possible for correctness.

Exercise 2: Measurement

Perform an ISO-type measurement series using the workload created in Exercise 1.

Exercise 3: Migration

Part 1
Migrate the workload created in Exercise 1 to another multi-user operating system (for instance to WINDOWS-XP, a proprietary UNIX system or a mainframe system). For migration rules see Section 12.11.

Part 2
Perform an ISO-type measurement using the migrated workload. For a SUT use a suitable IP system having the operating system specified in Part 1 .

Solutions
For solutions see file
`CD/Solutions/Solutions-Section12-12.pdf .`

13 Organisation and management of an ISO-type measurement project

For a performance measurement project all general rules of a project organisation and project management are valid. In this chapter some recommendations are made with special reference to the ISO method.

13.1 Deciding on the goals of the measurement project

Two decisions have to be made. The first decision is of whether the project deals with performance measurement or with software run time efficiency measurement. The second decision is the choice of the method to be used. Here it the choice is clear; measurement according to ISO/IEC 14756.

The following sections 13.1.1 and 13.1.2 describe points that have to be clarified in stating the goals of the project. They have to be recorded either on paper or electronically.

13.1.1 List of performance measurement goals

0. Attribute to be measured: Data processing performance

1. Method to be used: As defined in ISO/IEC 14756

Note: These two entries seem to be obvious, but they are not so for a reader not involved in the project. ■

2. Description of the SUT:
This is a short description of hardware, operating system, additional system software, network etc. A detailed description has to be given in an appendix of the project paper. In many cases several SUTs have to be measured for a performance comparison. They must all be summarised here and specified in detail in the appendix.

3. Choice of the workload:
One of the ISO workloads or an existing individual ISO-type workload can be suitable. If none of them is suitable for the user entirety under consideration, a new ISO-type workload has to be specified (see Chapter 12). Then the creation of the workload is an essential part of the project. Or it might be that a non ISO-type workload is intended to be used. It has to be clarified whether the attributes of this workload are sufficiently similar to the ISO method. If so a conversion can be made (see Section 14.8). The conversion is then an essential part of the project.

4. Single measurement or series
Decide of whether a single measurement or a measurement series (see Section 8.1) shall be performed. For a measurement series, the way of modifying the SUT has to be defined. Usually the number of users has to be varied incrementally (see Section 8.3).

5. Categories of values required for the results
Decide which results are required. They can be, can be for instance

- performance values (i.e. the vector triple of P)
  or it can be
- the rating values (i.e. the vector triple R_{TH}, R_{ME}, R_{TI})
  or it can be
- the N_{max} value.
  But also it can be
- a specified conclusion derived from the measured values
  for instance the cost per timely served user.

13.1.2 List of software run-time efficiency measurement goals

0. Attribute to be measured: Software run time efficiency

1. Method to be used: As defined in ISO/IEC 14756

Note: These two entries seem to be obvious, but they are not so for a reader not involved in the project. ■

2. The reference environment:
Define the reference environment. It contains the following components
(compare Section 10.2 and Figures 10-2, 10-7):

- the WPS
- the software of the levels between the user entirety and the measurement level
- the reference software of the measurement level
- the software of the levels between the measurement level and the hardware
- all the hardware

Then define all data necessary for a ISO-type workload description.

It is important that the speed of the SUT hardware is sufficient to ensure that none of the rating values (L-values) is less than 1, i.e. the reference environment must satisfy all user requirements (see tips concerning this in the examples of Section 10.4). Slower hardware would not make sense.

It is recommended that the reference environment is summarised here. A complete detailed description has to be given in a appendix of the project paper.

3. Specification of the SW to be measured:
The SW to be measured has to be specified in detail (name, manufacturer, release, build etc.). It has to be provided either as source or executable. For compilation the compiler options shall also be specified. The software has to be tested for full real operation.

4. Categories of values required for the results:
Decide which results are required. They can be, can be for instance

- the triple of the I vectors (see Section 10.3.2)
  or it can be
- the $I_{maxuser}$ value (see Section 10.3.3).
  But also it can be
- a specified conclusion from the measured efficiency values, for instance the number of additional timely served users when installing the measured software
  or it can be the cost of the software per timely served user.

13.2 Defining the responsibilities

The persons responsible for all parts and phases of an ISO-type measurement have to be defined. The division into parts naturally depends on the project and the personnel planning it.

Examples of detailed responsibility are:

- General planning of the project
- Setting up the project schedule
- Detailed definition of the workload (in case of performance measurement)
  or
  detailed definition of the reference environment (in case of
  software efficiency measurement
- Providing the components to be measured (SUT or software)
- Providing the test bed (room, computers, reference software, ISO-type RTE, etc.)
- Performing the measurement
- Computing the performance values; checking correctness;
  computing and documenting the measure report and the final result;
  checking the accuracy of all results
- Controlling the costs and expenditure of the project
- Performing an audit (if so planned)

13.3 Assessing the costs of the project

Normally an ISO-type measurement project is not a matter of trivial costs and technical expenditure. It cannot be performed by one person in one hour. Therefore when planning the project it is necessary to estimate the resources, both technical and personnel, and other costs.

Likewise, it is necessary to define the planned quality of the results (for instance, a first rough assessment or thorough and detailed assessment) and related aspects to be aimed at. The quality of results includes the values of the statistic criteria as in directory

`wln/C6-SPAP/`

(see Section 11.1). In case of a measurement series, the quality of the result and the accuracy of N_{max} depend on the step width of the number of users in the series. The best possible precision of N_{max} is ± 1 . Where there is more than one user type N_{max} is $\pm x$ where $x > 1$ (compare Section 8.2). If, for instance, N_{max} is about 1000 it can be sufficient to determine N_{max} with a precision of ± 50 .

In addition, contingency planning is desirable for dealing with possible unexpected results. Suppose that the planned duration of the RI is one hour, then it could happen that the SUT is not enough stable and needs 10 hours. In this case a reduced measurement precision will result. For such a situation provisions have to be made as for instance by making a decision as follows: the RI should not be longer than 2 hours; if the RI would be longer then the reduced precision of the measurement result (when keeping the RI less than 2 hours) will be accepted.

A balancing of quality of results, technical expenditure, manpower and costs when planning the measurement project is strongly recommended.

13.4 The project schedule

When planning a project the setting up of a schedule is mandatory in all cases. In such a schedule it can be seen whether the planned results will be available at the expected time.

13.5 Making the workload available

For an ISO-type measurement the workload consists of a set of data and programs (see Chapter 2 and Section 11.1). This set of information has to be brought into the state of being operational for the actual SUT and RTE under consideration. Then it can be installed. It is a regrettable fact that often the RTE is not able to read the ISO format of the WPS. In these cases the WPS has to be converted to the RTE format.

If one of the workloads in Annex F of ISO/IEC 14756 is being used, it has to be checked for being operational on the SUT. If the operating system is not a UNIX-SVR4 operating system the workload has to be converted for testing and running on the other system (see Section 11.4).

When using an individual ISO-type workload developed by someone else, this workload also has to be checked to determine whether migration to the actual SUT is necessary (i.e. operating system, compilers, data base system; see also Section 12.11). An additional aspect is the ownership of the workload. If the new user is not the owner then licensing aspects may need to be clarified.

An additional and important issue is the question of security and confidentiality of the stored data, i.e. flat files and/or data base content, if it is part of the workload. The data may be real production data restricted to its original computer centre. These data have to be made anonymous first, necessitating considerable additional work.

Whenever a suitable workload is not available a new workload has to be developed (see Chapter 12). In this case the phase of providing the workload will be part of the measurement project for which sufficient time, resources and costs must be planned.

It is strongly recommended that for clearness and readability the workload and especially the WPS are written in the ISO format (see Section 11.1 and the ISO workloads in Annex F of ISO/IEC 14756, see directory `CD/iso14756-orig-workloads/`).

13.6 Making the SUT operational and tuning it

A precondition for performing a measurement is the full operative functionality of the SUT. Therefore the SUT and all programs and data of the workload have to be completely installed and tested. For software efficiency measurement, both the software to be measured and the reference software have to be installed and made fully operational. These actions have to be performed first manually for one real user and then for a small number of real users. For advanced testing with many users the RTE can be used for user emulation. The amount of manpower of this phase should not be underestimated. It can be as much as needed when installing and testing for normal live operation.

To obtain, in case of performance measurement, optimum results from a SUT it should be tuned. This tuning is restricted to the adjustable parameters of the operating system, system

software (as for instance the data base management system) and the networking. Neither the programs nor the data of the workload may be altered or tuned.

For software efficiency measurement, the software to be measured can be tuned, but the reference software may not be altered or tuned.

13.7 Choosing an ISO-type RTE having sufficient performance

The RTE has to realise fully and correctly all ISO features and functions defined in ISO/IEC 14756. This concerns not only the task generation and submission, but also the recording of the data in an ISO-type logfile, and the checking of the accuracy of the measurement and of the calculation of the performance and rating values. Be careful when the supplier of an RTE states "ISO-type RTE"; he may have bent the truth. The RTE should be certified to ISO/IEC 14756 by an authorised expert.

The necessary hardware speed of the RTE depends, on the one hand, on the type of the workload and, on the other hand, on the power of the SUT. If for instance the workload has long preparation times and CPU intensive tasks types an RTE of moderate hardware power can be sufficient. But if the workload has task types that are not very CPU intensive (as for instance OLTP systems) and there are many emulated users, the RTE may need a hardware that is more powerful than that of the SUT.

13.8 Performing the measurement

Let us assume that all the preparatory work, as described above, has been completed successfully. Both when performing a performance measurement (single run or a measurement series) and when performing a software efficiency measurement (single run or series), each run follows the list of steps as in Fig. 9-2 in Section 9.1 . The duration of a run is longer than the RI because of the stabilisation phase and the supplementary run. Typically, additional time is needed for restoring the so-called stored data of the workload to its original state for the next run. And usually there is a large amount of work for checking the correct operation of the SUT. A well designed run needs from a minimum of a quarter of an hour to several hours, depending on the SUT and the workload.

Total needed time results from the planned runs, but often additional test runs are necessary which must taken into account.

When performing a measurement series the following tips may be helpful:

- Always start the series with only one user, or the minimum number if there are several user types (see Section 8.2). This will give you basic information about the behaviour of the SUT.
- Then gradually increment the number of users. Check the L-values (rating) and the CPU loading and of the SUT. Also always check the CPU loading of the RTE to ensure that it is not overloaded.
- Derive from the current measurement a value of the total number of users for the next run and try to interpolate N_{max} .
- Continue proceeding in this way. When approaching N_{max} try a value slightly less than this value.

- Use small steps in incrementing the number of users when the `L`-values approach the limit of 1.
- Take note that a run with too many emulated users can cause a serious system crash of the SUT. The system and stored data may become corrupted. Repairing this damage can lead to using a vaste amount of time and manpower. Make precautions for this situation.

Each measurement run must be documented (see Sections 9.3.2 and 9.4.1).

13.9 Computation of the performance values and rating values

The computation of these values from the recorded logfile is typically performed by programs which are part of the RTE. The computation is typically done within a few minutes. Therefore it is a matter of course that those results are provided by the RTE at the end of each measurement run. No extra action is needed. Typically the results of all runs are displayed on a screen and stored by the RTE.

13.10 Audit

As ISO/IEC 14756 is a technical standard there are no regulations concerning an audit. It is a common discretionary decision of the partners of a measurement project whether an audit shall be realised or not. If for instance the measurement results are only needed for internal company use an audit will not be needed. But if the measurement results are planned to be published or used for evaluating tenders, a audit may be advantageous or even necessary. In this case all parties must agree to the content.

14 Miscellaneous aspects

14.1 Measurement using a real workload

It would be no problem at all to record all tasks submitted by a real user entirety to a data processing system and to set time stamps for all relevant events such as the start of think time (preparation time), time of submitting tasks and time of completion of each task. But a problem would arise of how to classify the jobs into task types. For solving this problem it would be necessary to record an extended logfile containing much additional information about each job. This file would then have to be analysed for determining the task types. Then each job would have be assigned to one of the found task types. After the analysis of the logfile in this manner (and additional work) the values of the m-tuples B and T_{ME} can be computed. For computing the m-tuple E assumptions would be needed about the response time requirements (i.e. execution time requirements) of each user with respect to each task type. Using these assumptions it would be possible to compute the values of E . But the quality of these values would be no better than "estimated" instead of than "actually measured".

These assumptions define a WPS and the reference performance P_{Ref} can be computed from it. But the quality of these values would also be no better than "estimated" instead of "really measured". From the determined values P and P_{Ref} the rating vector triple could be computed. Again: The character of these values is only "estimated".

An additional problem is that a real workload is not really reproducible for performing additional measurements with the same or another SUT. Naturally the recorded logfile could be used as a new input for repeating the session. But how could the number of users be varied (for instance for realising a measurement series) ? Many more problems arise. Among these: the reference performance P_{Ref} for rating is only a rough estimate.

Because of all these problems it was decided to not include the use of real workloads in ISO/IEC 14756.

14.2 Measurement using automated sample users

A method often used for answering the following question is the use of sample users. The question is whether the response times of a multi-user system or a computer network fulfil the user requirements. A sample user can be realised by a user simulator (RTE) which emulates only one user ("automated sample user"). It is logged onto the system, which is running in real operation, in addition to the real users. The number of sample users need not be restricted to one, but typically it is small. For instance the number can be about 1 per mill or 1 per cent of the total number of active users. Naturally it is possible to use RTEs which operate according to the ISO method (see Fig. 14-1). The logfiles recorded by these RTEs can be analysed according to the ISO method. The ISO type performance values and rating values can be computed. But all these values and the conclusions derived from them are only valid to the extent that the workload defined for the sample user matches that of the real users. This assumption is very poor and unreliable.

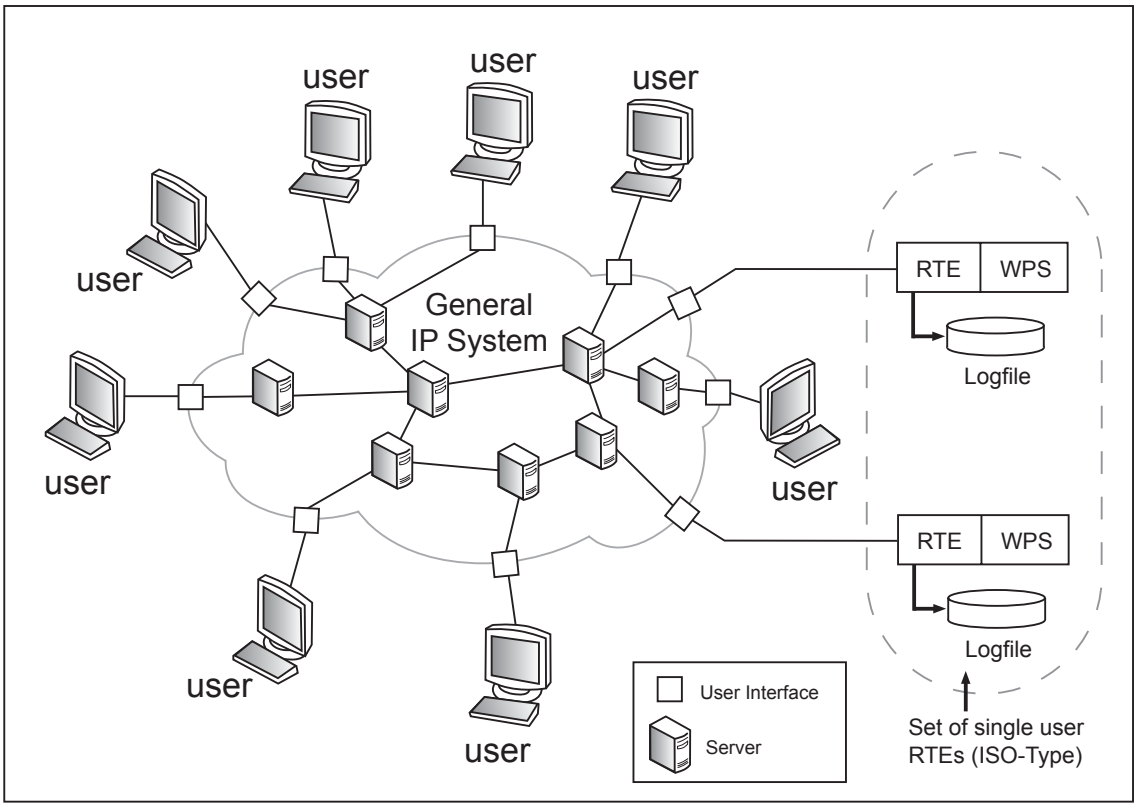


Fig. 14-1 Sample users

Using automated sample users may be a method for the rough monitoring of response times (execution times). But they are not suitable for determining reliably the ISO-type performance or the rating values of the IP system.

14.3 Measuring single-user systems

Though the ISO method was designed for measuring multi-user systems it can be applied also to single-user systems. The only one restriction is that the total number of users of the workload equals 1.

As there is only one user the derived performance measure N_{max} (see Chapter 8) cannot be used. Obviously the total number of users is fixed to 1. But measurement series which stepwise modify another parameter of the workload can be performed. For instance the ISO workloads COMPCENTER1 and COMPCENTER2 can be used by setting $N_{user}(1) = 1$ and varying another parameter, for instance the REP value. Hereof a new derived performance value can be determined, e.g. REP_{max}.

REP_{max} is the maximum REP value for which the 3*m R values are not less than 1. It is the maximum replication factor of the tasks for which the SUT fulfils the response time requirements of its single user.

Please note that the ISO workload COMPCENTER3 could not be used because the are more than 1 user types and $N_{tot} = 1$ impossible.

14.4 Hidden batch jobs in online transaction processing

When OLTP systems or OLTP computer networks are being measured a strange phenomenon often appears. With a small number of emulated users the execution times are typically short. With more users the mean execution times increase as expected and also the standard deviations increase. Increasing again the number of users increase the mean execution times. But the standard deviations increase unexpected strongly. Analysing the execution time distributions reveals a small number of tasks with execution times as much as 100 times the mean execution time. The task type which seems to be blocked changes randomly, a phenomenon that is not easily explained. No rule can be found.

A thorough investigation showed that there are some transaction task types that caused heavy CPU loading. For feigning short execution times these transactions are treated as completed and the keyboard (plus screen) is switched free for the next transaction. This is not explained in the user manual. But such a task (called "hidden batch job") is not completed at this time. It is changed to batch mode and placed to the background, hoping that some time later the CPU load will decrease sufficiently for completing the task. With a large number of active users this would probably not work. The hidden batch jobs would not be completed for a very long time and their computational results would be delayed. Tasks depending on these results would be blocked.

This problem was solved as follows. We recognised that these problematic transaction types have both external and internal task results (for "internal/external task result" see Section 3.4). Such a task is not completed at the appearance of the external result at the terminal (the screen message "ready" or similar) but at the arrival of the internal result, i.e. the completion of the hidden batch task. The moment of completion of those tasks is the appearance of the internal task result, i.e. the completion of the hidden batch job. Applying this definition of task completion to the problematic task types the SUT can no longer feign the too long execution times.

Correcting the problematic task type definitions in the WPS achieves that correct execution times are recorded and measured by an ISO-type measurement system. For these task types a new timeliness function has to be defined with a rather longer mean time. A good way of doing this is as follows: Try to determine the longest blocking time acceptable by the users and use this value as the time limit of the longest time class of the timeliness function. Do this for all task types that have hidden batch jobs.

14.5 A distributed RTE

Is it necessary to realise an ISO type RTE by use of only one computer ? The answer is "no". All users of an ISO-type workload work independently of each other. Therefore each user can be emulated by a separate computer. Any group of emulated users can share a computer. This is a distributed RTE. Only one special condition is necessary for a distributed RTE. This is a synchronisation mechanism for defining the common times t_0 (start time), t_1 (begin of the rating interval), t_2 (end of the rating interval) and t_3 (end of the supplementary run). For individual rating intervals the synchronisation mechanism has to be implemented according to the rules of this method as described in Section 6.2.1 . At the end of the measurement, i.e. after t_3, all the logfiles have to be collected up and evaluated. This is the responsibility of the computer that analyses them, performs the correctness tests and computes the performance values and the rating values.

The advantage of a distributed RTE is the emulation of higher number of users than possible with a single computer RTE. The disadvantage is that a synchronisation mechanism is necessary.

14.6 Life cycle of ISO-type workloads

The following is a list of aspects that influence the life cycle of an ISO type workload. The aspects are discussed below.

14.6.1 Workload definition and documentation

The data model, and the form and semantics of an ISO-type workload follow regulations which are accessible to everybody and available for a long time as they are in an international standard. Therefore the contents of a workload is on the one hand completely described and on the other hand needs no further comments or operating instructions (see Section 11.1). The workload file can be understood and used without additional explanations from the originator. An ISO-type workload is long-lived. This is contrary to traditional and proprietary benchmarks. Typically, only some parts or only the kernel of those benchmarks continue to be available. Only certain persons have the special knowledge and the data for running the benchmark. Again after some time those persons are no longer available and it would be impossible to understand this non ISO benchmark in detail or to run it again.

14.6.2 The RTE

The principles of an ISO-type measurement system are documented in the ISO/IEC 14756 and accessible to everybody. The RTE can be implemented on any multi-task multi-user operation system having a suitable network operability. Therefore the implementation is not dependent on the actual operating system available, the computer manufacturer, the hardware architecture and other details of the platform to be used. If there is the problem of lack of hardware speed, for instance for measuring a workload having a large number of users, a distributed RTE (see Section 14.5) can be used.

14.6.3 Type, architecture, manufacturer of the SUT

These properties do not influence an ISO-type workload, unless the workload was deliberately made to run only one particular configuration. But this would be contrary to the general intentions of the ISO/IEC 14756. Therefore an ISO-type workload can be used independently of the SUT architecture. This means that an ISO-type workload can in principle be used for measuring any existing IP system and also for new systems coming onto the market.

14.6.4 Power of the SUT

Most new computers and IP systems are more powerful than earlier machines. Consequently the execution times become shorter. Broadly said, this means that more users can be served. The usefulness of a workload is reduced when the SUTs becomes more powerful. A limit is reached when the execution times are too short even when the RTE emulates his maximum number of users. This number can be some thousands in case of a OLTP system workload, or some hundreds when the users access the SUT with telnet protocols. To resolve the problem parameters can be introduced to the activity types for adjusting the CPU load, the I/O loading and so on. For an example consider the `REP` factor of ISO workloads COMPCENTER1, COMPCENTER2 and COMPCENTER3 . When constructing a workload one of the goals should be longevity. Plan to introduce a factor for adjusting the weighting of the activity types without changing their general characteristics.

14.6.5 Applications contained in the workload

The applications have a strong influence on the life cycle of the workload. There are two classes.

- The first class comprises the day to day workloads. They are planned for short term use (for instance for an ad hoc comparison of some IP systems offered by suppliers to a customer). There are no special restrictions on the applications chosen for the workload.
- The second class comprises the workloads that are planned from the start for a long term use. Do not use low level languages. High level languages are more suitable for applications intended to be long-lived. Additionally the applications should not depend on short-lived special features of the operating system and system SW.

14.6.6 Final remarks

There are many additional aspects which could be discussed. From the five issues above it can be seen that the ISO method offers a good chance of making an ISO-type workload long-lived. This is contrary to most of the traditional performance measurement methods whose workloads short-lived. ISO-type workloads can, but must not, be long-lived.

14.7 Reliability aspects

When measuring the performance of an IP system or its software (run time) efficiency it is assumed that the system is in a stable operating state and has no reliability problems. With a system crash during the measurement (i.e. at any moment between t_0 and t_3) no measurement results become available. But what about the following situation? The system fails only for a short time, and starts again without having a loss of data or fails for several short periods ? Then the user entirety sees only some tasks having a long or a too long response time. An analogous situation happens when some user interfaces break down for a short time. In all these cases the measurement can run to a successful end. But the measured performance value $P = (B, T_{ME}, E)$ is worse than when the SUT runs quite normally. If the breakdowns last short enough the users cannot see if there were faults or if there was only a temporary lack of performance.

The important conclusion is: Short failures (but a not total breakdown) yield a reduction of the measured performance values. It is not possible to distinct between the effect of those failures and a real lack of performance. There is a grey area between normal operation and breakdown. This is only valid if there was no loss of data, i.e. all computational results of the SUT were correct. If not, then the measurement is not valid (see Section 4.1). Then the performance values should not be computed from the logfile because they would be meaningless.

14.8 Conversion of non ISO-type workloads to the ISO data model

14.8.1 Candidates for conversion

A vast number of programs exist that are supposed to measure performance. Many of them are not suitable for inclusion in a workload to be converted to the ISO data model. Programs are not suitable that indicate primarily or exclusively internal performance values (for instance CPU loading or storage usage). Also unsuitable are those programs that try only to determine the values referring to the reliability or accuracy of the IP system under consideration.

But suitable could be programs that check the speed of operation (see "third class" as described in Section 1.1). Many of those programs are only available as executable code without any description of the principles of task generation. There is no definition of the performance terms and no description of how they are calculated. These programs are not suitable for conversion to an ISO-type workload. Suitable candidates are only those programs where details such as task generation, the actual tasks, the performance terms and their calculation are clearly and completely described. All features should be fully documented and supported by source codes.

14.8.2 The conversion procedure

As the philosophies and methods of benchmarking vary widely a general and universally applicable conversion procedure is not possible. The basic idea is to use the procedure for of constructing an individual ISO-type workload (see Chapter 12).

Begin by investigating the benchmark for the tasks are submitted to the SUT. Then analyse as closely as possible which actions ISO-type users would perform when submitting the same job stream to the SUT as in the benchmark. Then follow the recommendations for constructing an individual ISO type workload (see Chapter 12). Usually the application software and data can be taken as it is (see Section 12.8). But sometimes it may have to be modified. Typically the benchmark does not offer suggestions for the timeliness functions so that they have to be based on your knowledge and experience.

When the workload has been converted an ISO type measurement can be performed. The ISO-type performance and rating values are determined. In nearly in all cases it would be pointless to try to express these results in the original terms of the benchmark. A possible exception is a benchmark that uses, in his original version, the number of timely served users or a related term for a performance measure.

14.8.3 Examples of conversions and sketches of some individual workloads

The following examples are only intended to give rough guidance; they are not guaranteed for either completeness of the conversion or accuracy.

14.8.3.1 The classic non-multiprogramming batch benchmark

Historically this is the oldest version of the batch benchmark. It was developed for SUTs having a non-multiprogramming operating system. The benchmark has only one activity type, the test program. Constructing an ISO-type workload yields a user entity consisting of just one user. He submits this activity to the SUT and repeats the submission when the program is completed. Therefore there is just one task type having the task mode 1. The timeliness function can be stated as simple as possible, for instance with only one time class having a suitable time class limit. For this limit we take the largest accepted run time of the test program, for instance 1 minute. The user's think time (preparation time) before submitting a task has no relevance. It can be set to an arbitrary value. For instance we can assume 1 second. These data produce the content of the directory `C1_WPS/` of the workload.

The contents of the directories `C2_AppPrgr/` , `C3_OSCP/` , `C4_Res/` , `C5_SData/` and `Prep/` result from the original benchmark description (for the names of the directories see Section 11.1). The directory `C6_SPAP/` should contain suitable values (as for instance used in the workloads shown in Annex F of ISO/IEC 14756). These parts are omitted here by reasons of shortness. They can be easily completed. This simple workload is shown in Fig. 14-2.

Measurement is simple. The duration of the RI would seem to be sufficient if it were long enough for the once only execution of the test program. But for reasons of being able to check the statistical significance of the execution time (see Section 4.3) the RI should be so long that it includes some more times the execution of the test program. The reason is as follows: most execution times will be nearly the same, but sometimes they may be quite different.

A) File General
(Short description and overview of the workload)

B) File C1 WPS/wps (workload parameter set)

1. Basic parameter values
(1) Total number of different user types: n = 1 ;
(2) Total amount of emulated users of each type: N_user(1) = 1
(3) Total number of different activity types: w = 1 ;
(4) Total number of different timeliness functions: p = 1 ;
(5) Total number of different task types: m = 1 ;
(6) Total number of different chain types: u = 1 ;

2. Activity type definitions

(1) Activity type number:	1
(2) The logical meaning of the input:	*)
(3) The length (number) of characters) of the input string:	3
(4) The input string itself:	run
(5) Activity type input variation:	none

*) Name of the shell script which has to be run.

3.Task type definitions

(1) Current number j of the task type:	1
(2) Number of the activity type:	1
(3) Value of the task mode M(j): :	1
(4) Type number of the timeliness function:	1

4.Definitions of the timeliness functions

(1) Order number of the timeliness function:		1
(2) Number of time classes of this function:		z=1
(3) z couples of values g_t and r_t, where g_t is the time limit and r_t is the maximum accepted relative frequency:	g_t(1):	1.0 min
	r_t(1):	1.0

5.Definitions of chain types

(1) The current number *l* of the chain type:	1
(2) The length L_chain(*l*) of the chain:	1
(3) The sequence of the task type numbers:	1

Fig. 14-2 ISO-type workload derived from the classic non-multiprogramming batch benchmark (abbreviated representation)

6.Definition of the chain probabilities q(i,*l*)

l	q(1,*l*)	i = number of user type
1	1.00	*l* = number of chain type

7.Preparation time mean values h(i,j)

j	h(1,j)	i = number of user type
1	1.0 sec	j = number of task type

8.Preparation time standard deviations s(i,j)

j	s(1,j)	i = number of user type
1	5.0 sec	j = number of task type

C) Directory C2_AppProgr
(…Programs…..)

D) Directory C3_OSCP
(…Operating system command procedures…..)

E) Directory C4_RES
(… Computational results…..)

F) Directory C5_SData
(…Stored data…..)

G) Directory C6_SPAP
File precision
DELTAq = ….. DELTAh = DELTAs = ……
File StatPar
ALPHA =…… drel = ……

H) Directory Prep
(….Implementation tips, directions of how to make the application software on the SUT ready for measurement run,…)

Fig. 14-2 (cont.) ISO-type workload derived from the classic non-multiprogramming batch benchmark (abbreviated representation)

Naturally checking computational correctness is mandatory (see Section 4.1). But checking the DELTA criterion (see Section 4.2) in this special case is of very little importance. Violation of the DELTA criterion would hardly influence the measured performance P values.

14.8.3.2 The classic multiprogramming batch benchmark

This benchmark typically uses more than one test program. The programs are started simultaneously on the SUT and run in parallel. When the longest running program finishes the test is stopped. Constructing the timeliness function we define only one time class. For the time class limit we take the longest acceptable run time of the test, for instance 200 seconds.

The classic multiprogramming batch benchmark runs on a single-user machine. Therefore the user entity consists of only one user. For starting the three programs at the same time we define three task types, TT_1 , TT_2 , TT_3 ; one for each program. All three task types have a preparation time mean value of zero and the task mode M = 0. This ensures that they are submitted without hesitation when started in a chain. All three task types have the same timeliness function. As their individual run times are meaningless (due to the trick shown below) we define the timeliness functions proper: One class with a time class limit of, for instance 100 minutes.

A little trick is necessary because the ISO method does not provide a synchronisation mechanism for the completion of two or more tasks. We write two little programs. The first one, called "sync", does nothing but to wait for the completion of TT_1 , TT_2 and TT_3 . The second one does nothing at all. It is completely a dummy program, called "dummy". We create two additional task types TT_4 and TT_5 . The activity type of TT_4 is the program "sync". Its task mode is M = 0. The activity type of TT_5 is the program "dummy". Its task mode is M = 1.

The next step is to define a chain type consisting of the sequence TT_1 , TT_2 , TT_3 , TT_4 , TT_5 . Starting this chain submits TT_1 , followed immediately by TT_2 , TT_3 and TT_4 (the synchroniser). But TT_5 does not start immediately. It waits for the completion of TT_4 (i.e. the moment when the last of the three test programs has delivered its computational result). Then, after a preparation time of for instance 1 second, TT_5 is submitted. It does nothing (i.e. has a nearly zero execution time). Its goal is only to avoid the start of a new chain before the actual one is finished. When TT_5 is completed the chain is completed and will be started again by the RTE, and so on. The timeliness function of TT_4 is important. It checks the longest run time of the three test programs with respect to not exceeding the 200 seconds run time as set above. For instance, we define for this timeliness function, two classes (200 seconds, 95%; 500 seconds, 100%). The timeliness function of TT_5 is not very relevant. We assign to it the same value as for TT_1 (or TT_2 or TT_3). These data produce the content of the directory C1_WPS/ of the workload.

The contents of the directories C2_AppPrgr/ , C3_OSCP/ , C4_Res/ , C5_SData/ and Prep/ result from the original benchmark description (for the names of the directories see Section 11.1). The directory C6_SPAP/ should contain suitable values (as for instance used

in the workloads shown in Annex F of ISO/IEC 14756). These parts are omitted here by reasons of shortness. They can be easily completed. This yields the workload as shown in Fig. 14-3.

Measurement is simple. The duration of the RI would seem to be sufficient if it were long enough for the once only execution of the chain. But for reasons of being able to check the statistical significance of the execution time (see Section 4.3) the RI should be so long that it includes some more times the execution of the chain. The reason is as follows: most execution times will be nearly the same, but sometimes they may be quite different.

Concerning the measured performance (and the rating) the task types TT_1 , TT_2 , TT_3 and TT_5 are not interesting. Only the run time of TT_4 is relevant.

Naturally checking computational correctness and is mandatory (see Section 4.1) and also checking the DELTA criterion (see Section 4.2).

14.8.3.3 The "Debit-Credit Test" for OLTP systems

This famous benchmark (see [ANON01] and [GRAY01]) was developed in about 1984 for measuring the performance of a large OLTP system of that time. It is one of the oldest predecessors of the present time system performance measurement methods (for the term "system performance measurement method" see Section 1.2). For reasons of simplicity this test includes only a single transaction type consisting of four steps ("update account", "write to history", "update teller", "update branch"). Before starting the transaction, 100 bytes of input data is read from the teller's terminal. After completing the transaction, 200 bytes of data are transferred to the terminal. The time from starting one transaction to the start of the next transaction is defined to be 100 seconds. The Debit-Credit test includes an early predecessor of the ISO timeliness function as follows: At least 95% of all transactions shall be completed within 1 second. But there is no upper limit. The number of active users in the measurement is increased as long as the cited condition ("95% tasks completed within 1 second") holds. This implies the idea of measurement series (see Chapter 8).

For reasons of brevity additional details of the Debit-Credit test are not described here. Many variations of the ISO WPS can be developed from this test. One variation is now shown.

We define only one activity type which executes the four steps ("update account", "write to history", "update teller", "update branch") in sequence without a pause. Using this activity type we define a single task type with the task mode M = 1. We include the typing of the 100 input bytes into the preparation time preceding the transaction. And we include the typing (the SUT of that time had a teletype terminal) of the 200 output bytes into the preparation time of the following transaction. We define a timeliness function with two time classes as follows: the first time class limit is 1.0 sec and the r-value is 0.95 (following the Debit-Credit specification). Because this same specification provides no data for the 2nd time class limit, we will set our own limit to 5 seconds. The r-value per definition equals 1.0 . The mean value of this timeliness function is

(0.95 * 1.0 sec) + (0.05 * 5.0 sec) = 1.2 sec .

The mean preparation time is therefore: 100 sec - 1.2 sec = 98.8 sec .

A) File General
(Short description and overview of the workload)

B) File C1 WPS/wps (workload parameter set)

1. Basic parameter values
(1) Total number of different user types: n = 1 ;
(2) Total amount of emulated users of each type: N_user(1) = 1
(3) Total number of different activity types: w = 5 ;
(4) Total number of different timeliness functions: p = 2 ;
(5) Total number of different task types: m = 5 ;
(6) Total number of different chain types: u = 1 ;

2. Activity type definitions

(1) Activity type number:	1	2	3	4	5
(2) The logical meaning of the input:	*)	*)	*)	**)	***)
(3) The length (number) of characters) of the input string:	5	5	5	4	5
(4) The input string itself:	Prog1	Prog2	Prog3	sync	dummy
(5) Activity type input variation:	none	none	none	none	none

*) Name of the shell script which has to be run.
**) Name of an executable which has to be run.
***) Name of an empty shell script which has to be run.

3.Task type definitions

(1) Current number j of the task type:	1	2	3	4	5
(2) Number of the activity type:	1	2	3	4	5
(3) Value of the task mode M(j): :	0	0	0	0	1
(4) Type number of the timeliness function:	1	1	1	2	1

4.Definitions of the timeliness functions

(1) Order number of the timeliness function:		1	2
(2) Number of time classes of this function:		z=1	z=2
(3) z couples of values g_t and r_t, where g_t is the time limit and r_t is the maximum accepted relative frequency:	g_t(1):	100 min	200 sec
	r_t(1):	1.00	0.95
	g_t(2):		500 sec
	r_t(2):		1.00

Fig. 14-3 ISO-type workload derived from the classic multiprogramming batch benchmark (abbreviated representation)

5.Definitions of chain types

(1) The current number *l* of the chain type:	1
(2) The length L_chain(*l*) of the chain:	1
(3) The sequence of the task type numbers:	1,2,3,4,5

6.Definition of the chain probabilities q(i,*l*)

l	q(1,*l*)
1	1.00

i = number of user type
l = number of chain type

7.Preparation time mean values h(i,j) and 8. Preparation time standard deviations s(i,j)

j	h(1,j)	s(1,j)
1	0.0 sec	0.0 sec
2	0.0 sec	0.0 sec
3	0.0 sec	0.0 sec
4	0.0 sec	0.0 sec
5	1.0 sec	0.0 sec

i = number of user type
j = number of task type

C) Directory C2 AppProgr
(…Programs, see original benchmark description …..)

D) Directory C3 OSCP
(…Operating system command procedures, see original benchmark description…)

E) Directory C4 RES
(… Computational results, see original benchmark description …..)

F) Directory C5 SData
(…Stored data, see original benchmark description …..)

G) Directory C6 SPAP
File precision | File StatPar
DELTAq = ….. DELTAh = DELTAs = ……| ALPHA =…… drel = ……

H) Directory Prep
(….Implementation tips, directions of how to make the application software on the SUT ready for measurement run …., see original benchmark description …)

Fig. 14-3 (cont.) ISO-type workload derived from the classic multiprogramming batch benchmark (abbreviated representation)

A) File General
(Short description and overview of the workload)

B) File C1 WPS/wps (workload parameter set)

1. Basic parameter values
(1) Total number of different user types: n = 1 ;
(2) Total amount of emulated users of each type: N_user(1) = x
The value of x will be set by the user of the workload.
(3) Total number of different activity types: w = 1 ;
(4) Total number of different timeliness functions: p = 1 ;
(5) Total number of different task types: m = 1 ;
(6) Total number of different chain types: u = 1 ;

2. Activity type definitions

(1) Activity type number 1:
---
(2) The logical meaning of the input: input to the 4 transaction steps
(3) The length (number) of characters) of the input string: 100
(4) The input string itself and (5) activity type:
Input variation, see original benchmark description

3.Task type definitions

(1) Current number j of the task type:	1
(2) Number of the activity type:	1
(3) Value of the task mode M(j): :	1
(4) Type number of the timeliness function:	1

4.Definitions of the timeliness functions

(1) Order number of the timeliness function:		1
(2) Number of time classes of this function:		z=2
(3) z couples of values g_t and r_t, where g_t is the time limit and r_t is the maximum accepted relative frequency:	g_t(1):	1.0 sec
	r_t(1):	0.95
	g_t(2):	5.0 sec
	r_t(2):	1.00

5.Definitions of chain types

(1) The current number *l* of the chain type:	1
(2) The length L_chain(*l*) of the chain:	1
(3) The sequence of the task type numbers:	1

Fig. 14-4 Proposal of the ISO-type workload derived from the Credit-Debit Test (abbreviated representation)

6.Definition of the chain probabilities q(i,*l*)

l	q(1,*l*)
1	1.00

i = number of user type
l = number of chain type

7.Preparation time mean values h(i,j)

j	h(1,j)
1	98.8 sec

i = number of user type
j = number of task type

8.Preparation time standard deviations s(i,j)

j	s(1,j)
1	0.0 sec

i = number of user type
j = number of task type

C) Directory C2 AppProgr

(…Programs, see original benchmark description …..)

D) Directory C3 OSCP

(…Operating system command procedures, see original benchmark description…)

E) Directory C4 RES

(… Computational results, see original benchmark description …..)

F) Directory C5_SData

(…Stored data, see original benchmark description …..)

G) Directory C6 SPAP

File precision

DELTAq = ….. DELTAh = DELTAs = ……

File StatPar

ALPHA =…… drel = ……

H) Directory Prep

(….Implementation tips, directions of how to make the application software on the SUT ready for measurement run, see original benchmark description …)

Fig. 14-4 (cont.) Proposal of the ISO-type workload derived from the Credit-Debit Test (abbreviated representation)

These data produce the content of the directory C1_WPS/ of the workload. The contents of the directories C2_AppPrgr/ , C3_OSCP/ , C4_Res/ , C5_SData/ and Prep/ result from the original benchmark description (for the names of the directories see Section 11.1). The directory C6_SPAP/ should contain suitable values (as for instance used in the workloads shown in Annex F of ISO/IEC 14756). These parts are omitted here by reasons of brevity. They can be easily completed. This yields the WPS as shown in Fig. 14-4.

The measurement is simple: Perform a measurement series (as explained in Chapter 8). Each measurement is to be performed according to the ISO rules. The duration of the rating period has to be sufficiently long, i.e. the statistic significance must be fulfilled. The performance value B(1) when N_{tot} equals N_{max} is the throughput which is used by the Debit-Credit test as the performance measure.

The WPS in Fig. 14-4 is a simple first version. A more detailed version could contain the following extensions: The 100 input bytes and the 200 output bytes each represent a separate activity. Assign them fitting timeliness functions. This yields two additional task types. Then define a chain type consisting of this three task types.

14.8.3.4 The SPECweb99 test for internet servers

The SPEC consortium (see http://www.specbench.org) published the intranet benchmark SPECweb99 in 1999 (see [SPEC01]). In a research project carried out by the Kassel University, (see [DIRLE04]) it was investigated whether the SPECweb99 test could be converted to the ISO workload data model. The result was yes. An ISO-type workload sketch is found in [DIRLE04] and also in directory

CD/workload-sketches/Web99/

of the CD as part of this book. It contains only the parts which were developed by the Kassel University performance laboratory, but for licensing reasons it does not contain application programs and data of SPEC. They are available directly from SPEC for holders of a license. The conversion project revealed a problem of the SPECweb99 test: There are too many chain types and many of them have very small q-values (below 1%). The least is $0.5*10^{-8}$. Therefore the according task types do not normally appear in a measurement of typical duration (for instance half an hour). Neither can the correctness of the execution of such a task be checked nor can its execution time be determined. Also the statistical significance of a measurement run will be not sufficient. These are not problems arising from the ISO method. But the conversion of the SPECweb99 benchmark to an ISO-type workload pointed out this problem. Up to this time it was unknown.

The solution was be found by redesigning the workload. Many of the infrequent chain types were replaced by a suitable new chain type. Its task types used input variation (see Section 2.7.1) in order to realise random access to data files. This solution was also used in the Kassel Internet Workload (see Section 14.8.3.5).

14.8.3.5 The KS-Web workload for intranet servers

The experiences made in the project described in Section 14.8.3.4 were used for designing an ISO type workload for intranet servers (see [DIRLE06]). A short sketch of it is shown in directory

`CD/Workload-sketches/KS-Web/`

of the CD as part of this book. The WPS includes 3 profiles of the intranet user community (day time, evening and night). A profile is characterised by the numbers of users of the user types (see WPS). The basic workload (for this term see Section 8.2) has N_{tot} = 10 users. This is the minimum increment when increasing the user numbers in a measurement series. The rule of having no q-values less than 0.05 (see Section 12.5) is obeyed.

14.8.3.6 The Kassel data warehouse workload

This workload concerns a so-called external data warehouse. Opposite to a traditional data warehouse, that delivers decision support for the company itself, the external data warehouse is a service of a company for its customers. In this example the company is a tax consultant. The data warehouse is a service for his clients. The directory

`CD/Workload-sketches/ExternalDWH/`

of the CD as part of this book is not the complete workload but only a short sketch.

There are 9 services, see file

`CD/Workload-sketches/ExternalDWH/C1_WPS/chains.txt .`

Each of them is represented by a task chain. Each activity type is shortly outlined by a scenario, see file

`CD/Workload-sketches/ExternalDWH/C1_WPS/Scenarios.txt .`

Typical for the user entirety is that the users order only a few inquiries per hour even in the rush hour. This yields long preparation times especially of task type number 1 (login). For WPS and comments see file

`CD/Workload-sketches/ExternalDWH/C1_WPS/wps .`

Using this WPS the measurement duration is long. Especially when only a small number of users are emulated several hours are needed for getting statistical significance. It is typically 10 or 20 hours. A measurement will therefore mostly started in the evening and can run over night.

For using summarised preparation times and execution times see file

`CD/Workoad-sketches/ExternalDWH/C1_WPS/times.txt .`

14.8.3.7 An ERP workload

Enterprise resource planning (or ERP) is an industry term for application software systems that are designed to serve and to support multiple business functions. An ERP software system can include functions as for instance manufacturing, order entry, accounts receivable and

payable, purchasing and transportation. The workload described here supports 4 applications:

- Financials and accounting (abbreviated AF)
- Materials management (abbreviated AM)
- Production planning (abbreviated AP)
- Sale (abbreviated AS)

These applications use a common database via a data base system. Employees always use only OLTP transactions of one application not of more applications. User type 1 is an employee using AF, user type 2 uses AM, user type 3 uses AP and user type 4 uses AS. The basic workload (for this term see Section 8.2) has the following user numbers:

$$N_{user}(1) = 4$$
$$N_{user}(2) = 3$$
$$N_{user}(3) = 1$$
$$N_{user}(4) = 2$$

The N_{tot} value equals 4 + 3 + 1 + 2 = 10 . This is the minimum increment when increasing the user numbers in a measurement series. In each application 5 or 6 (of many) OLTP transactions are chosen for representatives. There are 21 representatives. Each step of an transaction type is a task type. The steps of a OLTP transaction type make up a chain type. There are 21 chain types and 110 task types. Despite of this high number are no problems concerning too small q-values. This is due to the fact that a user only uses the chain types of his own application (see WPS). The rule of having no q-values less than 0.05 (see Section 12.5) is obeyed. Therefore the rating interval has an acceptable measurement duration for getting statistical significance. About one or two hours are mostly sufficient (when N_{tot} is about 100 to 200 users).

A sketch of the workload (not the complete description) is found in directory

`CD/Workload-sketches/ERP-WL/`

in the CD as part of this book.

An extension of the ISO method has proven to be useful for this workload. For getting (additional to the ISO rating) a summarising rating of the timeliness the following procedure was applied. After finishing the measurement and performing the ISO rating the records of the logfile are sorted according to the timeliness function of the task type. This yields three classes of records. The execution times of each class are processed according to Section 5.4.2 in order to get the number of timely served tasks of each class. Hereof a timely throughput value of each class is computed (using equation (5.9) of Section 5.4.1). Counting the number of tasks in each class and dividing it by the (mean) duration of the rating interval yields the throughput value of the class. Applying equation (7.12) of Section 7.6 produces a timeliness rating value for each of the classes. Details when using the Urn method are omitted here. These 3 timeliness rating values proofed to be useful for a fast summarising rating of the timeliness of the ERP system.

14.9 Example structure of an ISO-type measurement system

This section roughly outlines an example structure that originates from the DEMO implementation on the CD as part of this book. The RTE in DEMO has pregenerated task lists (no dynamic generation, see Section 3.2). It uses the Urn Method (see Chapter 6). DEMO uses a UNIX-SVR4 operating system.

14.9.1 The example structure

The steps of an ISO type measurement were explained in Section 9.1 . A flowchart of the measurement procedure is given in Fig. 9-2 . There are 10 steps. Although it could be possible to write some tools, steps 1 to 4 cannot normally be automated. The steps 5 to 10 can be automated by a set of computer programs. Such a software system is an ISO-type measurement system.

The ISO-type measurement system example consists of 11 modules, named M1, M2, M3, M4, M5, M6-1, M6-2/1, M6-2/2, M6-3, M7, M8.

The input to a measurement is found in the directories `C1_WPS/` to `C_6SPAP/` of the workload (see Section 11.1). The directories `C2_AppPrgr/` , `C3_OSCP/` and `C5_SData/` contain programs and data that have to be installed on the SUT. The values of `C1_WPS/` , `C4_Res/` and `C6-SPAP/` are input data of the RTE. The example structure of the measurement system is shown in Fig. 14-5 .

All modules of the measurement system, except M2 and M4, are independent of the type, manufacturer and operating system of the SUT. They can be written in any programming language, preferably a high level language.

The module M4 depends on the operating system of the SUT and on the communication protocols used at the interface between RTE and SUT. If the SUT uses another operating system the module M4 has to be rewritten and to be adapted to this operating system. And - naturally - if the communication protocols used by the workload differ from the above cited ones the module M4 has also to be adapted. M2 depends on the operating system of the SUT and has to be rewritten when the operating system of the SUT changes.

Note: The actual implementations of M2 and M4 of DEMO refer to a SUT having a UNIX SVR4 operating system. The used communication protocols are "telnet" and "ftp" and some others. ■

This yields a modular architecture as follows: For each SUT, a module M4 has to be written according to the operating system of the SUT and communication protocols of the interface between SUT and RTE. All other modules, except M2 discussed above, can be used without alteration. The adaptation of M2 is a simple task, as may be seen in Section 14.9.2 below.

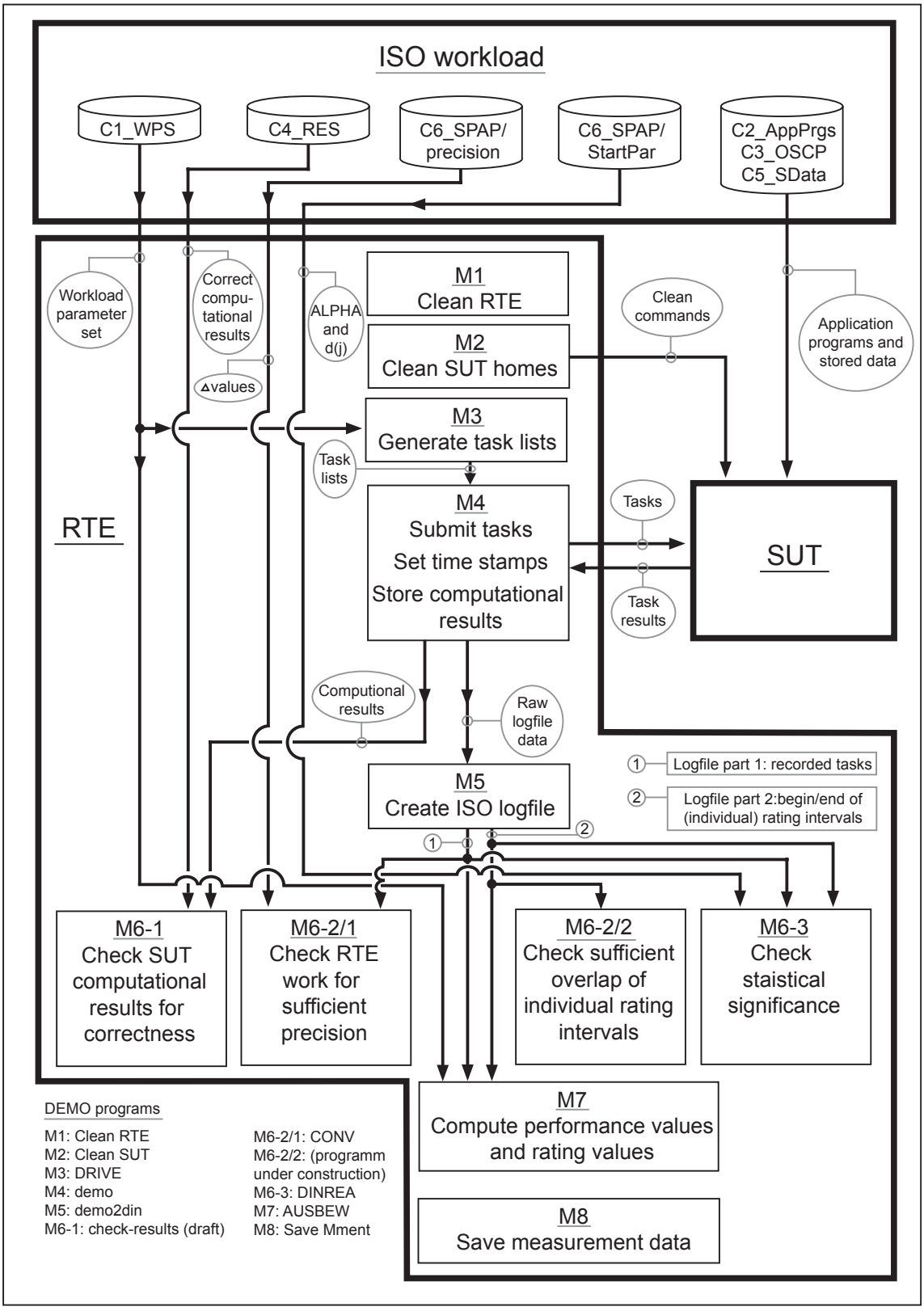


Fig. 14-5 Example structure of an ISO-type RTE

For an actual measurement the fitting modules M2 and M4 have to be inserted in the measurement system. All other modules are ready for use without changes. This modular architecture was used by the Kassel University performance laboratory. For Kassel's advanced ISO measurement system (not DEMO), modules M2 and M4 were implemented for, among others, the following SUT operating systems: UNIX-SVR4 (several dialects such as SUN-Solaris, Siemens-SINIX, IBM-AIX, HP-UX, LINUX), WINDOWS-NT, Siemens-BS2000, IBM-MVS, Univac-OS2000.

14.9.2 Short descriptions of the modules

A short explanation of each module and some general comments the implementation in the DEMO system are given here. For remarks on the actual implementation of DEMO see Section 14.9.3.

The modules have to be executed in the following sequence.

M1
This module deletes all unnecessary files from the home directory of the user running the measurement system (measurement operator). It consists of a little program, named "`CleanRTE`".

M2
This module deletes all unnecessary files from the home directories of the users which have an account on the SUT, and which are to be emulated by the RTE. It consists of a little program, named "`CleanSUT`".

M3
This module creates a set of three task files for each user. The first file contains the task list of the stabilisation phase. The second file contains the task list of the rating interval. The third file contains the task list of the supplementary run. The module consists of three programs named "`DRIVE`", "`dedula`" and "`dedula2demo`". (The names are historical and have no special significance.) They have to be executed in this sequence.

The Urn Method is implemented by program `DRIVE` . As ISO/IEC 14756 does not mention the Urn Method, it does not offer a sample program for this method of task list generation. In contrast, Annex E of the standard offer sample programs for the algorithms in some of the following modules.

M4
This module processes the task lists. Its kernel will be started in parallel for each user being emulated. If there are N_{tot}=4 users in the WPS, it will run `4` times parallel. Each process executes task lists, first its stabilisation phase (StP), then in its rating interval (RI) and finally in its supplementary run (SR). It also records the ISO logfile for the user he is emulating.

M4 consists of the program "`demo`". (The name is also historical.) "`demo`" is a simple implementation (see Section 14.9.3, "Comment 8"). There is no mechanism for checking whether the duration of the SR is fitting. This has to be controlled manually by the

measurement operator when running the program "DRIVE" (see module M3). The Kassel University ISO measurement system has, of course, automatic control of the SR. For simplicity it was omitted in DEMO.

M5
This module merges the individual logfiles of all emulated users and creates the ISO-type logfile of the measurement. The module consists of the program "demo2din". (Again, the name is historical.) In the actual implementation of DEMO the ISO logfile is represented by two parts: file "DIN.DAT" and file "ZEIT". DIN.DAT contains one record for each task executed as specified in ISO/IEC 14756. ZEIT contains, for each user, the begin time (t_{1ind}) and the end time (t_{2ind}) of its individual RIs (see Section 6.2.1). Individual RIs are necessary for the urn method used by DEMO (see module M3).

M6-1
The purpose of this module is to check the correct working of the SUT (see Section 4.1). It compares the computational results produced by the SUT during the individual RIs with the correct results stored in the directory C4_RES/ of the workload. M6-1 consists of the program "check-results". The program is written for workloads COMPCENTER1 and COMPCENTER2. For another workload it has to be modified accordingly.

M6-2/1
This module computes the actual task chain frequencies, and the actual preparation time mean values with their actual standard deviations from those logfile entries which belong to the individual RIs. The computed values are checked against the DELTA values as defined in the workload (in file C6_SPAP/precision). This checks for correct work of the RTE (see Section 4.2).

M6-2/1 consists of the program with the historical name "CONV". Its input is the file DIN.DAT (part 1 of the logfile). It computes the q_{meas} , h_{meas} and s_{meas} values. These values have to be checked, for practical demonstration, manually against the DELTA values as defined in the workload (see Section 4.2).

M6-2/2
This module checks if the individual RIs have sufficient overlap (see Section 6.2.4). For practical demonstration DEMO contains no program for this purpose. Checking has to be done manually on the screen, against the values in file "ZEIT" (part 2 of the logfile), displayed with an editor.

M6-3
This module performs the sequential test for statistical significance of the measurement results (see Section 4.3). It consists of the program "DINREA", (again a historical name). DINREA uses for input the files "DIN.DAT" (part 1 of the logfile) and "ZEIT" (part 2 of the logfile, the begin and end times of the individual RIs). For a better understanding of this program you are recommended to study intensively Appendix B.6 and the sample programs in Appendix E.6 of ISO/IEC 14756 and in [DIRLE02]. DINREA uses the fast computation procedure as described in Section 4.3.4.

M7
This module consists of the historically named program "AUSBEW-ENG". It computes performance values and rating values (see chapters 5 and 7) and writes a report of the computed results and on the measured data. The module has four parts.
Part 1 computes the performance value $P = (B, T_{ME}, E)$, see Chapter 5. For better understanding of this part you are recommended to study intensively Appendix B.3 and the sample programs of Appendix E.3 of ISO/IEC 14756 and in [DIRLE02]. They show the somewhat sophisticated algorithm of computing the values of E .
Part 2 computes the performance value $P_{Ref} = (B_{Ref}, T_{Ref}, E_{Ref})$ of the theoretical reference machine from the values of the WPS (see Sections 7.2 and 7.3). For better understanding of the algorithms you are recommended to study intensively Appendices B.1 and B.2 and the sample programs of Appendices E.1 and E.2 of ISO/IEC 14756 and in [DIRLE02]. The sample programs show the slightly complex algorithm for computing T_{Ref} .
Part 3 computes the rating vector triple (R_{TH}, R_{ME}, R_{TI}). This part is easy to understand (see Chapter 7).
Part 4 prints a listing of the measurement data including the values of the WPS.

M8
This module saves the data of the measurement run to the hard disk of the RTE for documentation. It consists of the program "SaveMment".

14.9.3 Some comments on the actual implementation of DEMO

For the actual release of DEMO see file
CD/DEMO-20/release.txt
of the CD as part of this book.

Comment 1
DEMO is not intended to be a tool for performing professional measurements, but only to demonstrate the ISO method and to provide a better understanding of it. It is not guaranteed to be free of errors or to correct results. But DEMO can show how to implement an ISO-type measurement system.

Comment 2
DEMO is under GNU license. It is intended to be a starting point. Everybody is encouraged to make improvements. The author of this book would be pleased to receive feedback from readers making improvements.

Comment 3
DEMO is written for an ISO measurement system using a LINUX platform and for SUTs also using LINUX as an operating system.

Comment 4
DEMO can be used by manually starting each module (in the sequence described in Section 14.9.2). This mode is intended for training and exercises. It allows the inspection of all files and data created by a module. DEMO can also be used via the shell procedure "runMeasurement" which is more convenient.

Comment 5
DEMO has three versions of the module M4 (task submission). Version 1 is intentionally simple ("simple demo"). It can execute only tasks which are a UNIX command or a call of a shell. Version 1 of M4 is not really an external user emulator. So as to be simple and robust the kernel of "simple demo" is only a UNIX shell running on the SUT itself. Therefore it is more like an internal emulator. But practical experience showed that "simple demo" mostly produces acceptable precisely results. Except for the fact that "simple demo" is not really external there are no further restrictions to the full functionality of an ISO type RTE. Small exceptions are reported in Comments 8 and 9. The interface between RTE and SUT uses "telnet" and "ftp" for protocols. It can emulate a maximum of 99 users. For Version 2 and 3 of M4 see below.

Comment 6
Version 2 of M4 is a real external user emulator. The interface between RTE and SUT uses "telnet" and "ftp" for protocols. Contrary to Version 1 ("simple demo") it is only an experimental version. It has not been properly tested. There are some problems with "telnet" when the total number is more than about half a dozen. Version 2 is not yet delivered on the CD as part of this book. An improved version is in preparation.

Comment 7
Version 3 of module M4 is intended for emulating users of an INTRANET. The communication protocol is http (hyper text transfer protokoll). Contrary to Version 1 ("simple demo") it is only an experimental version. It has not been properly tested. Version 3 is not yet delivered on the CD as part of this book. An improved version is in preparation.

Comment 8
For reasons of simplicity and for historical reasons DEMO does not implement all details of ISO/IEC 14756 functionality for automatic operation. Also for reasons of simplicity DEMO has no mechanism for automatically checking whether the duration of the SR was suitable. The duration has to be set manually by the measurement operator when running the program "DRIVE" (see module M3). The operator has to ensure that the SR is sufficiently long. A proposed improvement of DEMO (among others) is an extension of the module M4 for checking whether all tasks of the RI are completed by t_2 or t_{2ind}. The DEMO would then be able to terminate automatically the SR.

Comment 9
1. DEMO does not automatically check the tasks of the SR for the correct computation of their results and for keeping the `DELTA` criterion. These checks have to be performed manually. A proposed improvement of DEMO (among others) is to implement these checks.
2. DEMO omits the sixth entry in the logfile (sequential number of the task within its chain), see Section 3.4 . A proposed improvement of DEMO (among others) is to add this entry to the logfile. Additionally the correct complete sequence of each chain in the logfile should be checked. Also a proposed improvement of DEMO (among others) is to implement this functionality.

14.10 Applicability of the ISO method for measuring component performance

The ISO method was developed for measuring the performance of a complete IP system. There is an interesting question. Is the basic idea of the ISO method applicable for measuring the speed of a component of an IP system ? If yes, then is there a second question. Is there a possible benefit of such a measurement result ?

The first question, on applicability, can be answered as follows. The component being measured, for instance a hard disk or a RAID array, has to be regarded as a black box. Its user entirety is the complete hardware environment. The interface between this black box and its user entirety is the system cable to the disk unit. The activities of the tasks submitted to the black box are the read and write operations. Using them for building activity types the task types can be defined. Also timely functions can be defined and the task mode M can be determined. A suitable test bed can be a suitable computer. A measurement can be performed to determine the access rates (total throughput B), the rates of timely executed access operations (E) and the mean access times (T_{ME}). Each of this three terms is specified for each of the task types, producing

$$P = (B(1),... B(m); T_{ME}(1),...T_{ME}(m); E(1),...E(m)) .$$

Also a detailed rating (in sense of the ISO method) is possible, which produces the R-values.

If, however the component being measured is the CPU, an analogous scenario of a task generating environment (user entirety) and a task executing unit (black box "CPU") can be defined. The activity types are the machine instructions. The basic idea of the ISO method can also be applied.

Alternatively the ISO method could be applied to a data network as a black box. The tasks are the data transportation operations.

The second question, on possible benefit, can be answered as follows. The application of the ISO method to the measurement of components is an improvement on present methods. The term "performance" would be defined meticulously and would be measured more precisely. This would be a benefit and an advantage for engineers developing and producing components of IP systems. But using the ISO method for measuring components would scarcely be a contribution to the following well known unsolvable problem: It is usually not possible to compute the system performance value(s) of an IP system from the performance value(s) of its components.

14.11 Short comparison of some other methods with the ISO method

Note 1: This section refers only to methods for measurement of system performance values (i.e. to system performance as defined in Section 1.2) and not to component performance. Therefore batch benchmarks, graphic interface card benchmarks, super computer CPU benchmarks, RAID storage benchmarks, etc. are not considered in this section. ■

Note 2: In the Sections 14.8.3.1 and 14.8.3.2 some aspects of batch benchmarks are explained. Additionally, it is pointed out here that a batch benchmark does not realise an external user emulator. It is an internal emulator contrary to the ISO method which uses an external user emulator. ■

Note 3: This comparison is not intended to be a comprehensive report. Much more it is only a selection of present day system performance measurement methods. ■

14.11.1 Incomplete list of commonly used system performance measurement systems

The title of this section is to be understood literally. Many benchmarks appear every year. Important ones may be among them. Many new published benchmarks disappear after a short time because they are too specialised even when they have good ideas or an exceptional good theoretical basis. Many of them disappear because they are ad hoc solutions for just short time problems. The author has selected the following list of systems in the hope that they may be of reasonably long-term importance.

a) Monolithic benchmark systems
(See Section 1.6, items 3 and 5)

1. Names: TPC-C, TPC-H, TPC-R, TPC-W
   Publisher: Transaction Processing Performance Council
   (http://www.tpc.org)

2. Names: SPECapc, JBB2000, jAppServer2001, jAppserver2002,
   MAIL2000,SFS97_R1, SPECweb99, SPECweb99SSl
   Publisher: Standard Performance Evaluation Corporation
   (http://www.specbench.org)

3. Name: OLTP
   Publisher: Fujitsu Siemens Corporation
   (http://www.fujitsu-siemens.de)

4. Names: NetBench, WebBench
   Publisher: Veritest, Division of LioNBRIDGE
   (http://www.veritest.com)

5. Name: BaanERPsuite (about 8 benchmarks)
   Publisher: SSA Baan Company
   (http://www. baan.com)

6. Name: Oracle Application Standard Benchmark Version 11
   Publisher: Oracle Corporation
   (http://www.oracle.com/apps_benchmark)

7. Names: SYSmark2004, webmark2004
   Publisher: BAPCO Corporation
   (http://www.bapco.com)

8. Name: Exchange Server Benchmark MMB2
   Publisher: Microsoft Corporation
   (http://www.microsoft.com)

9. Names: NotesBench, Server.Load
   Publisher: Lotus Notes Consortium
   (http://notesbench.org)

10. Names: SAP Application Benchmarks (about 18 benchmarks)
    Publisher: SAP Corporation
    (http://www.sap.com/benchmark)

b) General purpose measurement systems

11. Name: SilkPerformer
    Publisher: Segue Software Inc.
    (http://www.segue.com)

12. Name: LoadRunner
    Publisher: Mercury Interactive Company
    (http://www.mercuryinteractive.com)

13. Name: QALoad
    Publisher: Compuware Corporation
    (http://www.compuware.com)

14. Name: Robot
    Publisher: Rational
    (http://www-306.ibm.com)

15. Name: S_ATURN8000
    Publisher: Zott+Co Company
    (http://zott.net)

14.11.2 Short comparison

This comparison of the systems cited in Section 14.11.1 focuses on the principles used in the benchmarks. It does not deal with aspects of marketing, economics, costs of the benchmarks, etc. . The comparability of produced measurement results between the different methods is not considered. This is especially due to the fact that most benchmark developers make strong claims for the uniqueness of their measures and the incompatibility of their measured results to different benchmarks.

Sytems No. 1 to 10:
All of these systems are monolithic systems (see Section 1.6, items 3 and 5). I.e. the measurement method, the performance terms and the workload are developed and defined as one unit. The method cannot use a different workload type. The scenario of the workload, the workload parameters, the application programs of the SUT, etc., are a fixed set of interdependent elements. Only some parameters of the workload, for instance the number of active users or the size of the database, can be modified. Contrary to the ISO method, the workload is not described using a general purpose data model. One reason (among others) for this situation is probably that the main steps of the design of the benchmark were done before the ISO/IEC 14756 was published. Timeliness functions are not used or only used indirectly. One example of such an indirect (or incomplete) use is the TPC-C test. 95% of the response times may not exceed 2 seconds. But there is no upper limit. Contrary to the ISO method, which always uses an external user emulator, some of the methods use an internal emulator. The think times of the users (preparation times) do not vary randomly in all methods. Often they are constant. Some of the methods are "ready for run benchmarks" as used by the ISO method. Others are "high level benchmarks". Checks for the correct operation of the SUT or the RTE are realised only in a few of the methods. But, if there are checks, they are not so strict as in the ISO method. A check for the statistical significance was not found anywhere. Most of the methods define an auditing procedure (contrary to the ISO method which is restricted to technical aspects). The performance measures of the methods are very different and do not agree with those of the ISO method. Mostly, the measures are very simplifying. Simplifications are for instance: Use of the mean response time of all tasks and not differentiated to each task type; total maximum possible throughput of all tasks regardless of the execution time category. Each publisher of the methods emphasises that his performance terms are very different from all other methods. He also emphasises that the measured values cannot be converted to the terms of any method of another publisher or even to another of his own methods. The final aspect to be investigated is the rating. A rating of the measured performance to the user requirements is hardly ever realised (contrary to the ISO method). Only those measurement methods that use the maximum number of served users as a performance measure include indirectly a rating (example: TPC-C). This rating is not so strict as the ISO rating.

Systems No. 11 to 14:
These systems are not monolithic. They are able to use different workload types. The workload is typically created by tracing a real user session. The tracing produces so-called scripts. They are comparable to a sequence of task chains of the ISO method. Some parameters of the scripts, as for instance the think times, can be modified before being used by the mostly external user emulator. But the workload description method is not in accordance with the ISO workload parameter set. A workload defined by use of the ISO WPS cannot be

processed by the systems as an input to the RTE. There is no task generation according to the ISO method, neither in the sense of pre-generated task lists nor in the sense of dynamic generation. No timeliness functions, as defined by ISO, are used. The performance measures are simpler than those of the ISO method. A rating of the measured performance values with respect to the users performance requirements is either not implemented at all, or not in accordance with the ISO method. But most of the systems use a term as a performance measure which has the meaning of a maximum number of users served. Therefore, indirectly they include a rating. It is less satisfactory than the ISO rating. The correct working of the RTE and the SUT is usually checked somehow. But there is no checking for the statistic significance of the performance values.

System No. 15:
This system includes the complete implementation of ISO/IEC 14756. Additionally it contains tools for determining an ISO WPS from tracing a real user session. The system is developed for professional use, also with a large number of emulated users and all types of application software. It is able to use a wide spectrum of protocols of the interface between the RTE and the SUT (also INTRANET/INTERNET network protocols). It is not an experimental system.

14.12 Applying ISO/IEC 14756 to Function Point Measurement (Written by Eberhard Rudolph)

14.12.1 Overview

Function Points are a normative unit of measure of software size. Function Points are derived from elementary function types contained in software user requirements of computer systems. The principles of Function Point are defined by ISO/IEC 14143-1 [ISO14143]. There are several different Function Point metrics. Detailed guidance in deriving Function Point results can be found in ISO/IEC 19761 [ISO19761] and [IFPUG01].

Traditionally the use of Function Points has been restricted to the development and maintenance of computer software. Typical applications would be estimation of software development time or determining the productivity of software development. A detailed summary of conventional Function Point applications can be found in [JONES01].

Rudolph and Dirlewanger [RUDOL01] proposed to expand the use of Function Points to quantify the consumption of software. Using the principles of ISO/IEC 14756 they demonstrated the feasibility of correlating activated Function Points with the consumption of computing resources.

ISO/IEC 14756 uses tasks (or user transactions) as basic form of software delivery. The Function Point technique provides a quantitative measure for such tasks.

14.12.2 Activated Function Points (AFP)

Using established Function Point metric concepts it is proposed to measure the use of software by Activated Function Points (AFP). An Activated Function Point is a Function Point of a function type (or task in ISO/IEC 14756 terminology) performed for the user.

Monitoring (or estimating) the consumption of Function Points provides a new form of management information. It will assist capacity planning, resource consumption estimation, analysis of the effectiveness of operations management and evaluation of the performance of delivery platforms. In addition, Function Point consumption measurement will enable service suppliers to charge in a user understandable form, based on functionality delivered rather than computer resources consumed.

14.12.2.1 Deriving AFP

In a first step the function types contained in the software to be measured are identified and classified using the counting rules of the established Function Point techniques. Each time a transaction function type (for the IFPUG method a transaction function type EI, EQ, or EO) is executed its Function Point value is added to the AFP value.

ISO/IEC 14756 is based on tasks. Therefore only transaction (task) function types can be considered by AFP. The ISO/IEC 19761 FFP method (see [ISO19761]) is based on transactions and does not contain any other function types. Therefore all FFP functions will be considered in an AFP count.

The IFPUG Function Point method distinguishes between transaction and data function types. AFP will only include transaction function types (EI, EO and EQ). The data function types (ILF and EIF) of the IFPUG method have to be excluded. Indirectly data function types are already considered in the transaction type functions. When calculating the Function Points of a transaction function type the FTR (File Type Referenced) consideration largely determines the value.

The provision of IFPUG data function types (particularly ILFs) can be measured by alternative metrics such as number of records (or instances) stored.

14.12.2.2 AFP example

Let's assume a simplistic Human Resources application has just one single EI, EQ and EO function. Let's further assume that the FP value of the EI is 4 FP, of the EQ is 3 FP and of the EO is 7 FP. If during one hour the users performed 20 EIs, 10 EQs and 3 EOs then there would be counted:

```
20*4 + 10*3 + 3*7 = 131 Activated Function Points (AFP).
```

The Human Resources application delivered 131 AFP during that hour.

14.12.3 Using AFP with ISO/IEC 14756

Rudolph and Dirlewanger introduced AFP to the ISO method as follows: In a first step the Function Points of the SUT were identified and counted. For a defined user entity the WPS is to be determined. Rudolph and Dirlewanger then propose to perform the following series of measurements in the ISO test bed:

Starting from one the number of users is increased until the maximum number of users N_{max} is obtained which can be served timely by the measured IT system. In case of $n > 1$ user types the measurement series starts from the minimum increment for number of users (see Section 8.2). All other values of the WPS remain unchanged.

The throughput values (i.e. the transaction rates) in this situation are named $B_{max}(1)$, $B_{max}(2)$,.... for the first, second,.... transaction type. Let be $FP(1)$, $FP(2)$,.... the function point value of the first, second,.... transaction type. The AFP value corresponding to the j-th transaction type is:

$$AFP(j) = B_{max}(j) * FP(j) \quad . \qquad (14.1)$$

The total rate of AFP is:

$$AFP_{tot} = AFP(1) + AFP(2) + ... AFP(m) \quad . \qquad (14.2)$$

where m is the number of transaction types. The average rate of AFP flowing per user is:

$$AFP_{av} = AFP_{tot} / N_{max} \quad . \qquad (14.3)$$

If needed the individual AFP rate $AFP_{user}(x)$ of the defined user x can be computed similarly from the measured transaction rates in the N_{max} case.

It is important to note the following points:

1. AFP_{tot} is the maximum rate of AFP which can be served "timely" by the SUT.
2. When increasing the number of users beyond N_{max} then the IT system will probably produce an AFP rate greater than AFP_{tot}. But the users will have unsatisfying long response times (execution times). I.e. the service levels required by the users are no longer fulfilled.
3. When decreasing the number of users below N_{max} the system will operate below accepted peak performance.

14.12.3.1 SAP-R/2 measurement

To demonstrate the feasibility of the proposed approach Rudolph and Dirlewanger [RUDOL01] used a major SAP R/2 test performed 1994 in a mainframe environment. This example also demonstrates the possibility to apply the AFP analysis to existing ISO/IEC 14756 test results. Using the task specifications for the selected R/2 the AFP results could be derived from the original measurements.

The standard SAP-R/2 test system used was developed by highly experienced SAP staff. The measurement was originally performed in 1994. The mainframe used was an IBM ES9121-190 with 8 Gartner-MIPS and 64 MB main storage. The operating system was MVS-ESA 4.3.0.

For details of the experiment and its environment see [DIRLE05].

Fig. 14-6 lists the components of the SUT. Four modules of the SAP-R/2 system were used: Financial Accounting (RF) 40% of the users, Materials Management (RM-MAT) 30% of the users, Production Planning and Control (RM-PPS) 10% of the users and Sales and Distribution (RV) 20% of the users. There were 4 user types, one for each SAP module. The total number of transaction types was 15. Note that RM-PPS transactions TD41 and TL06 were amalgamated since they did always run together.

Application	*Transaction*	*Frequency (%)*	*Think Time*	*Timeliness (sec)*	*Type*	*FP*
RF	TB01	30	90	20	EI	6
	TB03	10	45	10	EQ	4
	TB05	25	75	16.5	EI	6
	TB14	25	50	10	EQ	6
	TS21	10	50	10	EQ	4
RM-MAT	TE21	20	100	23	EI	6
	TE24	20	45	10	EQ	4
	TL01	20	60	13	EI	4
	TL11	20	60	13	EI	6
	TR01	20	80	23	EI	6
RM-PPS	TD41 + TL06	40	70	56	2 EI	10
	TD43	30	65	13.2	EQ	3
	TD56	30	60	10	EO	7
RV	TA01	50	115	33	EI	6
	TB03	50	55	10	EQ	4

Fig. 14-6 SAP-R/2 example

The think time (preparation time) shown in Fig. 14-6 is in seconds. The Function Point assessment was done without access to the detailed functional description of the individual SAP R/2 transactions. The FP results were obtained from the input data elements. The referenced logical files were derived from the input data and it was assumed that an access control file was used with all transactions.

The timeliness requirements in this measurement were more complex. Each R/2 transaction was broken down to several internal steps. There were three types of steps defined. Their timeliness functions are:

a) Simple data entry : 90% < 2 sec, 100%< 4 sec; avg. 2.2 sec
b) User dialog : 60% < 4 sec, 100%< 8 sec; avg. 5.6 sec
c) Background job : 80% < 30 sec,100%< 60 sec; avg. 36.0 sec

Each transaction could contain several of each these steps. In most cases 2/3 of the steps were of type "a)" and 1/3 of type "b)". The timeliness (mean) values of the individual transactions shown in the table (i.e. T_{Ref} value of the transaction) are the rounded sums of the individual steps of the transaction.

The ISO type performance measurement found the maximal number of users N_{max} = 110 users.

The duration of the measurement is set by the ISO procedure ensuring that statistically significant results are produced. In this experiment the duration of the measurement was 25.24 minutes. In this time 11404 AFP were processed. The throughput was

$$AFP_{tot} = 451.82 \text{ AFP/min.}$$

The average rate of AFP flowing per user is

$$AFP_{av} = 4.11 \text{ AFP / min and user.}$$

In this experiment the number of used SAP R/2 transaction types was intentionally taken low in order to limit the amount of manpower and cost of the experiment. There are no principal limitations to include in the ISO user model a large number of transaction types, to represent them in the WPS, and to perform the measurement.

14.12.4 Limitations

Function Points do not consider the size of business data. In some instances when correlating AFP with the use of computer resources the size of application files may have an impact. An external output function type, such as the printing of a telephone directory, may require considerably less computing resources for a small telephone listing file of a few hundred listings compared with the same function having to produce a directory for a million entries. Special consideration is required when using AFP in a workload containing a considerable amount of transactions which process complete business files rather than just a few records.

14.12.5 Opportunities

Quantifying the SUT using Function Points enables the ISO/IEC 14756 standard to provide comparative performance data. Applied software measurement techniques (see [JONES01]) already available and used for the development of software, can be extended to include the delivery of software functionality.

Appendix A: CD as a part of this book

1. Foreword
   See file `CD/Foreword-of-this-CD.txt`

2. Contents
   File `CD/Contents-of-this-CD.txt`

3. GNU General Public license
   See file `CD/GNU-gpl.txt`

4. ISO-IEC 14756 original workloads
   See directory `CD/iso14756-orig-workloads/`

5. Logical steps of the OSCPs of the ISO computer centre workloads
   See file `CD/Supplement-to-ISO14756.pdf`
   Contents:
   - Workload COMPCENTER1
   - Workload COMPCENTER2
   - Workload COMPCENTER3

6. Two ISO workloads converted for LINUX SuSE 9.1

6.1 Workload COMPCENTER1
   See directory `CD/Linux-workloads/CC1-Linux9/`
   Installation:
   see file `CD/Linux-workloads/Install-CC1-Linux9.txt`

6.2 Workload COMPCENTER2
   See directory `CD/Linux-workloads/CC2-Linux9/`
   Installation:
   see file `CD/Linux-workloads/Install-CC2-Linux9.txt`

7. Sketch of the ISO workload COMPCENTER1,
   converted for NT 4.0
   See directory `CD/NT-workloads/`

8. Sketches of some ISO type individual workloads
   See directories
   `CD/Workload-sketches/ERP-WL/`
   `CD/Workload-sketches/ExternalDWH/`
   `CD/Workload-sketches/KS-Web/`
   `CD/Workload-sketches/Mainframes/`
   `CD/Workload-sketches/Web99/`

9. Measurement system DEMO 2.0 (implemented for LINUX SuSE 9.1)
9.1 Manual
See directory `CD/DEMO-20/DEMO-manual/`
9.2 Software
See directory `CD/DEMO-20/DEMO-sw/`
9.3 XDEMO
See file `CD/DEMO-20/XDEMO.txt`

10. Detailed documentation of a measurement using DEMO
See directory `CD/Mexample/`

11. Solutions of exercises
See directory `CD/Solutions/`

12. Files of the exercises
See directory `CD/Sol-files/`

Note 1: This compact disc was created using a LINUX operating system. If using another operating system for reading it is not guaranteed that correct data are obtained or displayed, but "`.txt`" files, "`.email`" files and files named `readme` or `README` can be displayed also by WINDOWS Editor or WordPad . "`.pdf`" files can be opened by ACROBAT reader. ■

Note 2: For extracting the "tar" archives
see file `CD/Contents-of-this-CD.txt` . ■

References

[ALLEN01] Allen, A. O.: "Introduction to computer performance analysis with Mathematica", AP Professional, Harcourt Brace & Company Publishers, Boston, 1994, ISBN 0-12-051070-7

[ANON01] Anon, et al: "A measure of transaction processing power", DATAMATION, International Edition, Issue April 1985, pp. 112 - 118

[BOLCH01] Bolch, G.: "Performance evaluation of computer systems using queuing models", B. G. Teubner Publishers, Stuttgart, Germany, 1989, ISBN 3-519-02279-6 (German title: "Leistungsbewertung von Rechensystemen mittels analytischer Warteschlangenmodelle")

[DIN01] German National Standard DIN 66273: "Measurement and rating of data processing performance",
Part 1: "Measurement and rating method",
Part 2: "Standard workload type A",
Part 3: "Standard workload type B",
Part 4: "Standard workload type C",
Beuth-Verlag GmbH, Burggrafenstraße 6, 10772 Berlin,
Germany, 1991 - 2002. This standard is written in German language.
(German title: "Messung und Bewertung der Leistung von DV-Systemen".)
An English translation of Part 1 is available by DIN:
Paper UDC 681.3:53.08:003.62 .

[DIRLE01] Dirlewanger, W.:"A remote terminal emulator for measuring DP performance, based on a new definition of data processing performance", DAS RECHENZENTRUM, Issue 1/1986, pp. 19 -24, C. Hanser Publishers, Munich, Germany, ISSN 0343-317X, (German title: "Ein Treiber zur DV-Leistungsmessung auf der Basis eines neuen Leistungsbegriffes")

[DIRLE02] Dirlewanger, W.: "Measurement and rating of data processing performance", Huethig Buch Verlag GmbH, Heidelberg, Germany, 1994, ISBN 3-7785-2147-0 (German title: "Messung und Bewertung der DV-Leistung auf Basis der Norm DIN 66273".)

[DIRLE03] Dirlewanger, W.: "The DIN method, a new approach for measuring rating data processing performance", Proceedings IEEE International Computer and Dependability Symposium IDPS 96 Illinois, pp. 200 - 209, IEEE Computer Society Press, Los Alamitos, CA, 1996, ISBN 0-8186-7484-9

[DIRLE04] Dirlewanger, W.: "ISO type representation of the internet workload SPECweb99", Technical report (July, 20th, 2001), University of Kassel, Dpt. of Mathematics and Computer Science, working group Performance Measurement, 34132 Kassel, Germany, Heinrich Plett-Straße 40.
(For a copy see directory `workload-sketches/Web99/` in the CD which is part of this book.)

[DIRLE05] Dirlewanger, W.: "Performance Measurement using a SAP workload, applying a new measurement method for mainframes", Proceedings of the Annual Meeting of the Computer Measurement Group Central Europe (CMG-CE), paper Nr. 7, pages 1 - 29, May 1995, Königwinter. (German title: "DIN-Leistungsmessung mit einer SAP-Last, Erprobung der neuen Methode im Mainframe-Bereich".)

[DIRLE06] Dirlewanger, W.: "A intranet workload, described by the ISO workload model", Technical report (July, 29th, 2001), University of Kassel, Dpt. of Mathematics and Computer Science, working group Performance Measurement, 34132 Kassel, Germany, Heinrich Plett-Straße 40. (For a copy see directory `workload-sketches/KS-Web/` in the CD which is part of this book.)

[GRAY01] Gray, J.: "The benchmark handbook for database and transaction processing systems", Morgan Kaufmann Publishers, San Francisco, CA, 1993, ISBN 1-55860-292-5

[IFPUG01] "IFPUG Function Point counting practices manual, Release 4.2", International Function Point User Group, 191 Clarksville Road Princeton Junction, NJ 08550, USA, 2005.

[ISO14143] International Standard ISO/IEC 14143-1:1998 "Information technology Software measurement - Functional size measurement - Definition of concepts". ISO/IEC Copyright Office, Case Postale 56, CH 1211, Genève 20, Switzerland, 1999

[ISO14756] International Standard ISO/IEC 14756: 1999:(E), "Information technology – Measurement and rating of performance of computer-based software systems", ISO/IEC Copyright Office, Case Postale 56, CH 1211, Genève 20, Switzerland, 1999

[ISO19761] International Standard ISO/IEC 19761: 2002, "Software Engineering – COSMIC-FFP: Functional size measurement method", ISO/IEC Copyright Office, Case Postale 56, CH 1211, Genève 20, Switzerland, 1999

[JAIN01] Jain, R.: "The art of computer systems performance analysis; Techniques for experimental design, measurement, simulation and modelling", John Wiley & Sons Inc., New York, 1991, ISBN 0-471-50336-3

[JONES01] Jones, C. "Applied software measurement: assuring productivity and quality", McGraw-Hill, New York, 2nd edition, 1996, ISBN 0-07-032826-9

[RUDOL01] Rudolph, E. and Dirlewanger, W. "Applying Function Points to the delivery of IT services", International Function Point User Group, 191 Clarksville Road Princeton Junction, NJ 08550, USA, IFPUG 2002 Annual Conference, San Antonio, TX, 24-27 Sept 2002, pp.197-226.

[SPEC01] "SPECweb99 Benchmark", Standard Performance Evaluation Corporation (SPEC), 6585 Merchant Place, Suite 100, Warrenton, VA 20187, USA

Abbreviations

AFP = activated function points

APS0 = reference application software

APS1 = the actual application software to be measured for software run time efficiency

ASCII = American standard code for information interchange

AT = activity type

C = the programming language called "C"

CD = compact disc read only memory

chap. = chapter

COBOL = the Common Business Oriented (programming) Language

COM = computer output on microfilm

COSMIC = Common Software Measurement International Consortium

CPU = central processing unit

CT = chain type

DEMO = a demonstration system of the ISO measurement method

EI = external input (a transaction function type)

EIF = external interface file (a data function type)

EO = external output (a transaction function type)

EQ = external inquiry (a transaction function type)

eq. = equation

ERP = enterprise resource planning

FFP method = full function point method

Fig. = figure

FORTRAN = the programming language "formula translator"

FP = function point

FTR = file type referenced

GFLOPS = giga floating operations per second

GNU license = Open Source Foundation software license

http = hyper text transfer protocol

HW = hardware

IFPUG = International Function Point User Group

ILF = internal logical file (a data transaction function type)

I/O = Input/Output

IP system = information processing system

IP_0 = reference system for software efficiency measurement

IP_1 = the actual IP system when measured software run time efficiency

ISO = International Standardisation Organisation

JAVA = the platform-independent object-oriented programming language

LATEX = text system for natural sciences

LINUX = the "GNU-licensed" UNIX type operating system
developed by Linus Thorwaldsen

MIPS = million instructions per second

OLTP = online transaction processing system

OP = observation period

OSCP = operation system command procedure

PC = personal computer

PDF = Adobe encrypted text

PS = post script

RAID = redundant array of independent disks

REP = replication factor of "activities" in a task

RI = rating interval

RTE = remote terminal emulator

sect. = section

SR = supplementary run

SS-nU = non-UNIX type system software

SS-U = UNIX type system software

StP = stabilisation phase

SUT = system under test

SW = software

TEX = text system for natural sciences

TF = timeliness function

UNIX = the "open" multi-user operating system

UNIX-SVR4 = UNIX system V release 4

WPS = workload parameter set

Symbols

(symbol – description – reference section(s))

a(j,*l*)	number of tasks of the j-th task type within a chain of the *l*-th type, 7.3.2
AFP_{av}	average rate of activated function points, flowing per user, 14.12.3
AFP(j)	activated function point value corresponding to the j-th task type, 14.12.3
AFP_{tot}	total rate of activated function points, 14.12.3
AFP_{user}(x)	individual rate of activated function points of user x, 14.12.3
ALPHA	confidence coefficient, 2.9.3, 4.3.2
AT_j	j-th activity type, 2.3
B	throughput vector, 5.1, 5.2
b	same as b(j) but index j omitted for shortness, 5.4.2
B(j)	throughput of the j-th task type, 5.1, 5.2
B_{max}(j)	throughput of the j-th task type in case of N_{tot} equaling N_{max}, 14.12.3
b(j)	total number of tasks of the j-th task type submitted during the rating interval, 5.2
B_0	throughput vector of the reference system IP_0 for software run time efficiency measurement, 10.3.2
B_0(j)	throughput of the j-th task type of the reference system IP_0 for software run time efficiency measurement, 10.3.2
B_1	throughput vector of the system IP_1 the SW efficiency of which is to be measured, 10.3.2
B_1(j)	throughput of the j-th task type of the system IP_1 the software run time efficiency of which is to be measured, 10.3.2
B_{ind}(j)	throughput of the j-th task type in the individual rating interval, 6.2.2.1

$b_{ind}(j)$ total number of tasks of the j-th task type submitted during the individual rating interval, 6.2.2.1

$b_{ind}(v,j)$ total number of tasks of the j-th task type submitted by the v-th user during its individual rating interval, 6.5.1

$B_{ind}(v,j)$ throughput of tasks of the j-th task type submitted by the v-th user during its individual rating interval, 6.5.1

B_{Ref} reference vector of throughput, 7.2, 7.3.2

$B_{Ref}(j)$ reference value of throughput of tasks of the j-th task type, 7.3.2, 7.2

$b_{RefClass}(k)$ reference value of total number of tasks in the k-th time class, 5.4.2

CT_l l-th chain type, 2.4

d half width confidence interval, general, 4.3.2, 2.9.3

d(j) half width confidence interval of the preparation time mean value of tasks of the j-th task type, 2.9.3

d_{rel} relative half width confidence interval, 2.9.3

$DELTA_h$ same as $DELTA_h(i,j)$ in case of being uniquely for all user types and all task types, 2.9.2, 4.2.3

$DELTA_h(i,j)$ maximum tolerated relative difference of the mean preparation times of tasks of the j-th task type to the required mean preparation time; this term refers to the tasks submitted by the users of the i-th type, 4.2.3

$DELTA_q$ same as $DELTA_q(i,l)$ in case of being uniquely for all user types and all chain types, 2.9.2, 4.2.2

$DELTA_q(i,l)$ maximum tolerated relative difference of the relative chain frequency of chains of the l-th chain type to the required relative chain frequency; this term refers to the chains submitted by the users of the i-th type, 4.2.2

$DELTA_s$ same as $DELTA_s(i,j)$ in case of beeing uniquely for all user types and all task types, 2.9.2, 4.2.4

$DELTA_s(i,j)$ maximum tolerated relative difference of the standard deviation of the mean preparation times of tasks of the j-th task type to the required standard deviation; this term refers to the tasks submitted by the users of the i-th type, 4.2.4

$DIFF_h(i,j)$ measured relative difference (absolute value) of the mean preparation times of tasks of the j-th task type to the required mean preparation time; this term refers to the tasks submitted by users of the i-th type, 4.2.3

$DIFF_q(i,l)$ measured relative difference (absolute value) of the relative chain frequency of chains of the *l*-th chain type to the required relative chain frequency; this term refers to the chains submitted by users of the i-th type, 4.2.2

$DIFF_s(i,j)$ measured relative difference (absolute value) of the standard deviation of the mean preparation times of tasks of the j-th task type to the required standard deviation; this term refers to the tasks submitted by users of the i-th type, 4.2.4

E timely throughput vector, 5.1, 5.4.1

e same as e(j) but index j omitted for shortness, 5.4.2

E(j) timely throughput of the j-th task type, 5.1, 5.4.1

e(j) total number of timely tasks of the j-th task type submitted during the rating interval, 5.4.1, 5.4.2

E_0 timely throughput vector of the reference system IP_0 for software run time efficiency measurement, 10.3.2

$E_0(j)$ timely throughput of the j-th task type of the reference system IP_0 for software run time efficiency measurement, 10.3.2

E_1 timely throughput vector of the system IP_1 the software run time efficiency of which is to be measured, 10.3.2

$E_1(j)$ timely throughput of the j-th task type of the system IP_1 the software run time efficiency of which is to be measured, 10.3.2

$E_{ind}(j)$ timely throughput of the j-th task type in the individual rating interval, 6.2.2.3

$e_{ind}(j)$ total number of timely tasks of the j-th task type submitted during the individual rating interval, 6.2.2.3

$E_{ind}(v,j)$ timely throughput of the v-th user with respect to tasks of the j-th task type in his individual rating interval, 6.5.3

$e_{ind}(v,j)$ total number of timely tasks of the j-th task type submitted by the v-th user during his individual rating interval, 6.5.3

E_{Ref} reference vector of timely throughput, 7.2, 7.3.2

$E_{Ref}(j)$ reference value of timely throughput of tasks of the j-th task type, 7.2

$f(l,k)$ task type of the k-th task in a chain of the l-th type, 7.3.2

FP(j) function point value of the j-th task type, 14.12.3

$g_T(k)$ time class limit of the k-th time class of a timeliness function, 2.5, 5.4.2

h(i,j) Preparation time mean value of the tasks submitted by a user of the i-th type before submitting a task of the j-th task type, 2.7.6

$h_{meas}(i,j)$ mean value of measured preparation times of tasks of the j-th type; this term refers to the tasks submitted by all users of the i-th type, 4.2.3

$\underline{hx}$ random variable of preparation time of tasks of the x-th task type, 3.1

$I_{maxuser}$ software run time efficiency value related to the number of timely served users, 10.3.3

$I_{ME}(j)$ software run time efficiency value related to mean execution time of the j-th task type, 10.3.2

$I_{TH}(j)$ software run time efficiency value related to throughput of the j-th task type, 10.3.2

$I_{TI}(j)$ software run time efficiency value related to timeliness of the j-th task type, 10.3.2

$L_{Chain}(l)$	length (number of tasks) of the l-th chain type, 7.3.2, 11.1
M	task mode, 2.3
M(j)	task mode of the j-th task type, 7.3.2
m	total number of task types, 2.3, 2.6
M*(i)	an intermediate term of the "beta-formula", 7.3.2
$m_m(N)$	mean value of N samples (general), 4.2.5
N	total number of samples taken from a random changing variable, 4.2.5
n	total number of user types, 2.6
N_{GCD}	greatest common divisor, 8.2
N_{max}	maximum number of timely served users, 8.1
N_{max0}	maximum number of timely served users of the reference system IP_0 for software run time efficiency measurement, 10.3.3
N_{max1}	maximum number of timely served users of the system IP_1 the when measuring the software run time efficiency, 10.3.3
N_{tot}	total number of users, 2.6
$N_{user}(i)$	total number of users of the i-th user type, 2.6
P	system performance, 5.1
p	total number of timeliness functions, 2.3, 2.6
P_0	Performance of the reference system IP_0 for software run time efficiency measurement, 10.3.1,.10.3.2
P_1	Performance of the system IP_1 when measuring the software run time efficiency, 10.3.1,.10.3.2
P_{Ref}	reference performance, 7.2
q(i,l)	relative frequency of using the l-th chain type by an user of the i-th user type, 2.7.5

$q_{meas}(i,l)$ measured relative frequency of chains of the l-th type; this term refers to the chains submitted by all users of the i-th type, 4.2.2

REP replication factor, 11.3.1

REP_{max} maximum value of the replication factor of a single user system for serving the user timely, 14.3

R_{ME} mean execution time rating vector, 7.5

$R_{ME}(j)$ mean execution time rating value of the j-th task type, 7.5

$r_T(k)$ relative time class frequency of the k-th time class of a timeliness function, 2.5, 5.4.2

R_{TH} throughput rating vector, 7.4

$R_{TH}(j)$ throughput rating value of the j-th task type, 7.4

R_{TI} timeliness rating vector, 7.6

$R_{TI}(j)$ timeliness rating value of the j-th task type, 7.6

$s(i,j)$ preparation time standard deviation of tasks of an user of the i-th type with respect of submitted tasks of the j-th task type, 2.7.7

$s_m(N)$ standard deviation of N samples (general), 4.2.5

$s_{meas}(i,j)$ standard deviation of measured preparation times of tasks of the j-th type; this term refers to the tasks submitted by all users of the i-th type, 4.2.4

$sum_{ET}(j)$ sum of all execution times of tasks of the j-th task type, 6.5.2

t_0 begin of the stabilization phase, 3.3, 9.1

t_1 begin of the rating interval, 3.3, 9.1

t_{1ind} same as $t_{1ind}(v)$ but index v omitted for simplicity, 6.2.1, 9.1.5.2

$t_{1ind}(v)$ begin of the individual rating interval of the v-th user, 6.5.1

t_2 end of the rating interval, 3.3, 9.1

t_{2ind} same as $t_{2ind}(v)$ but index v omitted for simplicity, 6.2.1, 9.1.5.2

$t_{2ind}(v)$ end of the individual rating interval of the $v-th$ user, 6.5.1

t_3 end of the supplementary run, 3.3, 9.1

t_{3ind} end of the supplementary run after the individual rating interval, 6.2.3, 9.1.5.2

t_4 end of the supplementary run in case of using individual rating intervals, 6.2.3, 9.1.5.2

$t_{ET}(j,x)$ execution time of the $x-th$ task of the $j-th$ task type, 5.3

$t_{ETind}(v,j,x)$ execution time of the $x-th$ task of the $j-th$ task type in the individual logfile of the $v-th$ user, 6.5.2

TF_i $i-th$ timeliness function, 2.3, 2.7.3

T_{ME} mean execution time vector, 5.1, 5.3

$T_{ME}(j)$ mean execution time of the $j-th$ task type, 5.1, 5.3

T_{ME0} mean execution time vector of the reference system IP_0 for software run time efficiency measurement, 10.3.2

$T_{ME0}(j)$ mean execution time of the $j-th$ task type of the reference system IP_0 for software run time efficiency measurement, 10.3.2

T_{ME1} mean execution time vector of the system IP_1 when measuring the software run time efficiency, 10.3.2

$T_{ME1}(j)$ mean execution time of the $j-th$ task type of the system IP_1 when measuring the software run time efficiency, 10.3.2

T_{MR} mean duration of the individual rating intervals of all users, 6.2.4

T_R duration of the rating interval, 5.2

T_{Ref} reference vector of mean execution times, 7.2, 7.3.1

$T_{Ref}(j)$	reference mean value of execution time of the `j-th` task type, 7.2, 7.3.1
T_{Rind}	duration of the individual rating interval, 6.2.1
$T_{Rind}(v)$	duration of the individual rating interval of the `v-th` user, 6.5.1
T_{Sind}	duration of the individual supplementary run after the individual rating interval, 6.2.3
TT_j	j-th task type, 2.3
`u`	total number of chain types, 2.4, 2.6
`v`	current user number, 6.5.1
`var(N)`	variance (i.e. the square of the standard deviation) of `N` samples, 4.3.2
`w`	total number of activity types, 2.3, 2.6
x_i	`i-th` sample of a random changing variable, 4.2.5
$x_{ME\text{-}lower}$	lower bandwidth value for extended ISO type mean execution time rating, 7.7.2
$x_{ME\text{-}upper}$	upper bandwidth value for extended ISO type mean execution time rating, 7.7.2
$x_{TH\text{-}lower}$	lower bandwidth value for extended ISO type throughput rating, 7.7.2
$x_{TH\text{-}upper}$	upper bandwidth value for extended ISO type throughput rating, 7.7.2
$x_{TI\text{-}lower}$	lower bandwidth value for extended ISO type timeliness rating, 7.7.2
`Y`	total number of generated preparation times (urn method), 6.3.3
`z`	total number of time classes of a timeliness function, 2.5
Δ	preparation time step (urn method), 6.3.3
■	end of note (general)

Index

(Item - corresponding section(s))

A
accepted measurement error - 4.3.2, 8.4, 9.5
activated function points - 14.12.2
activity - 2.3
activity type - 2.3
activity type definition - 2.7.1
advanced workload parameters - 2.9
assessing costs (of a measurement project) - 13.3
audit - 13.10, 1.6

B
basic parameters (of a workload) - 2.6
basic workload - 8.2
batch benchmark -14.8.3.1, 14.8.3.2
batch job - 3.2
batch mode - 2.3, 11.5.1.1, 11.5.2.1, Section 2.6 in file CD/Supplement-to-ISO14756.pdf
benchmark - 1.1, 1.6
black box 1.3

C
chain - 2.4
chain type - 2.4
chain type definition - 2.7.4
check (of correct working of the RTE) - 4.2
check (of correct working of the SUT) - 4.1
check (of statistical significance) - 4.3
completion (of an ordered activity) - 2.3, 3.4
component performance - 1.2, 14.10
computational result - 2.8
confidence coefficient - 2.9.3, 4.3.2
confidence interval - 2.9.3, 4.3.2
conversion (of non ISO-type workloads) - 14.8
conversion procedure (for non ISO-type workloads) - 14.8.2
CPU utilisation 1.2
cycle (Urn Method) - 6.3.2.1
cycle index (program loop) - Sections1.1 to 1.5 and 2.1 to 2.5 in file in CD/Supplement-to-ISO14756.pdf

D
data warehouse workload - 14.8.3.6
Debit-Credit Test – 14.8.3.9
dialog mode - 2.3, 11.5.1.1, 11.5.2.1, Section 2.6 in file CD/Supplement-to-ISO14756.pdf
distributed RTE - 14.5
dynamic check (of computational results) - 4.4
dynamic task generation - 3.2

E
efficiency (of software run time) - 10.2
ERP (enterprise resource planning) workload - 14.8.3.7
executed instructions (number of) - 1.2
execution time (response time) - 1.6, 2.3, 3.4
execution time (response time) requirement – 1.6, 2.3, 2.5
external performance value 1.2
external task result - 3.4, 14.4

F
function point - 14.12.1
function point consumption - 14.12.2
function point measurement - 14.12
feedback - 2.1

G
goal of a measurement project - 13.1
greatest common divisor - 8.2, 12.5

H
hidden batch job - 14.4
hierarchical model (of an IP system) - 10.1
high level benchmark - 1.6

I
increment of user numbers - 8.2
individual rating interval - 6.1
individual rating interval (begin of) - 6.2.1
individual rating interval (end of) - 6.2.1
information processing system - 1.1
input variation - 2.7.1, 2.8
interactive job - 3.2
interactive mode - 2.3, 11.5.1.1, 11.5.2.1, Section2.6 in file CD/Supplement-to-ISO14756.pdf
internal performance value 1.2
internal task result - 3.4, 14.4
intranet workload - 14.8.3.4, 14.8.3.5
ISO workload data model - 2.2
ISO workloads - Chapter 11
ISO-type workload - 2.1
ISO-type workload definition - 2.2

J
job queue 1.2

K
KS-Web workload – 14.8.3.5

L
length of a task chain type - 2.2, 2.4, 2.7.4
life cycle of a workload - 14.6

logfile - 3.1, 3.4
look-ahead transformation 3.2

M
management (of an ISO measurement project) - Chapter 13
maximum number of timely served users - 8.1
mean execution time of j-th task type - 5.1, 5.3, 6.2.2.2
mean execution time rating - 7.5
mean execution time rating value - 7.5
mean execution time rating vector - 7.5
mean execution time vector - 5.1, 5.3
mean value - 4.2.5, 4.3.4
measure of performance - 5.1, 8.1
measure of software run time efficiency - 10.3.2, 10.3.3
measurement error - 4.3.2, 8.4, 9.5
measurement method (of computer performance) 1.4
measurement of component performance - 1.2, 14.10
measurement of system performance - 1.2, 14.10
measurement operator's protocol - 9.4.1
measurement report - 9.3
measurement result file (logfile) - 3.4
measurement series - 8.1, 10.3.3, 10.4.1.3, 10.4.2.3, 14.3
migration of an ISO-type workload - 11.4, 11.5, 12.11
minimum increment of user numbers - 8.2
modelling method 1.4
monolithic benchmark - 1.6, 14.11.1, 14.11.2
multi-processor - 1.1
multiprogramming 1.2
multiprogramming factor 1.2
multi-user - 1.1

N
NOWAIT mode - 2.3
number (total) of tasks of the j-th task type - 5.2
number (total)of timely executed j-type tasks - 5.4.1, 5.4.2

O
observation period - 3.3
online transaction processing - 2.1, 11.3.4
organisation of an ISO measurement project - Chapter 13
overall rating - 7.7
overlap (of individual rating intervals) - 6.2.4

P
performance - 1.1, 5.1
prediction method 1.4
pregenerated task list - 3.2, 6.3.2
preparation time (think time) - 2.3
preparation time mean value - 2.7.6
preparation time standard deviation - 2.7.7
project schedule - 13.4

R
random generation methods - 3.6
rating (of computer performance) 1.5, Chapter 7
rating interval - 3.3
rating interval (duration of) - 5.2
rating interval (estimation of the magnitude) - 12.7
ready for run benchmark 1.6
real workload - 14.1
reference environment - 10.2
reference IP system - 10.3.1
reference mean execution time - 7.2, 7.3.1
reference performance - 7.1
reference software - 10.2, 10.3
reference throughput - 7.2, 7.3.2
relative chain frequency - 2.7.5
relative confidence interval - 2.9.3
reliability - 1.1, 14.7
remote terminal emulator 1.6, 3.1
replication factor - 8.1, 11.3
reproducibility of measured results - 9.5
response time (execution time) - 1.6, 2.3, 3.4
response time (execution time) requirement - 1.6, 2.3, 2.5
responsibilities (of persons involved in a project) - 13.2
RI part of the logfile - 3.4, 3.5
run time efficiency - 1.6, 10.2

S
safe keeping period - 9.4.3
sample user - 14.2
sequential test - 2.9.3, 4.3.2, 4.3.3
simulation method 1.4
single-user system - 14.3
software efficiency term - 10.3.2, 10.3.3
software efficiency value (related to a task type) - 10.3.2
software efficiency value (related to N_{max}) - 10.3.3
software run time efficiency 10.2
SPECweb99 test – 14.8.3.4
SR part of the logfile - 3.4, 3.5
stabilisation phase - 3.3
standard deviation - 4.2.5, 4.3.4
statistical significance - 2.9.3, 4.3
storage utilisation 1.2
supplementary run - 3.3
system performance - 1.2, 5.1
system under test - 1.6

T
task - 2.2, 2.3
task chain - 2.2, 2.4
task chain type - 2.2, 2.4

task completion - 3.4
task list - 3.1
task mode - 2.3
task result - 3.4
task submission - 3.4
task type - 2.2, 2.3
task type definition - 2.7.2
test bed (performance measurement) - 9.1
test bed (SW run time efficiency) - 10.1, 10.3.1, 10.4.1.1, 10.4.2.1
theoretical reference machine - 7.2
think time (preparation time) - 2.3
think time (preparation time) mean value - 2.7.6
think time (preparation time) standard deviation - 2.7.7
threshold - 2.5
throughput of j-th task type - 5.1
throughput rating - 7.4
throughput rating value - 7.4
throughput rating vector - 7.4
throughput vector - 5.1, 5.2
time class - 2.5
time class limit - 2.5
timeliness function - 2.3, 2.5
timeliness function definition - 2.7.3
timeliness rating - 7.6
timeliness rating value - 7.6
timeliness rating vector - 7.6
timely reference throughput - 7.2
timely throughput of j-th task type - 5.1, 5.4.1
timely throughput vector - 5.1, 5.4.3
transaction (task type) - 14.12

U
urn method - Chapter 6
user behaviour parameter - 2.7
user community - 2.1
user entirety - 2.2, 1.5
user interface 1.3
user type - 2.6

V
validation - Chapter 4
variance - 4.3.2, 4.3.4
view of the workload (interface) - 2.1
view of the workload (system internal) - 2.1
view of the workload (user-oriented) - 2.1

W
WAIT mode - 2.3
workload - 1.6, 2.1, 2.2
workload data model - 1.6, 2.2
workload parameter set - 2.10

X

Y

Z
