This paper is an extract from *Project Control for Software Quality*, Editors, Rob Kusters, Adrian Cowderoy, Fred Heemstra and Erik van Veenendaal. Shaker Publishing, 1999. ISBN 90-423-0075-2.

Test point analysis: a method for test estimation

Erik van Veenendaal and Ton Dekkers

Abstract

This document describes the test estimate preparation technique known as test point analysis (TPA®[1]). TPA can be used to objectively prepare an estimate for a system test or acceptance test. TPA covers only so-called black-box testing; an estimate for the test activities, which precede black-box testing (i.e. white-box testing), will already have been included in the estimate produced by function point analysis. So, while the function point analysis productivity factor covers the white-box testing, it does not cover system testing or acceptance testing.

TPA can also be used if the test hour allowance has been predetermined. By performing a TPA, any risks involved can be clearly identified by comparing the objective TPA estimate with the predetermined number of test hours. With TPA, it is also possible to determine or calculate the relative importance of the various functions, with a view to using the available testing time as efficiently as possible.

1. Philosophy

When formulating an estimate for a black-box test, three elements are relevant: the *size of the information system* to be tested, the *test strategy* (selection of system components and quality characteristics to be tested and the coverage of testing) and the *level of productivity*. The first two elements together determine the volume of testing work to be undertaken (expressed in test points). If the number of test points is multiplied by the productivity (the amount of time needed to perform a given volume of testing work) you get a test estimate in hours. The three elements, (size, test strategy and productivity) are considered in more detail below.

1.1. Size

The first element to be considered is the size of the information system. For TPA purposes, the size of an information system is determined mainly by the number of function points [1] assigned to it. However, a number of additions or amendments need to be made, because certain factors which have little or no influence on the number of function points *are* pertinent to testing. The factors in question are the following:

- *Complexity;* complexity relates to the number of conditions in a function. More conditions almost always means more test cases and therefore a greater volume of testing work.
- *Interfacing;* the degree of interfacing of a function is determined by the number of data sets maintained by a function and the number of other functions, which make use of those data sets. Interfacing is relevant because these “other” functions will require testing if the maintenance function is modified.

[1] TPA is registered trademark from IQUIP

- *Uniformity;* it is important to consider the extent to which the structure of a function allows it to be tested using existing or slightly modified specifications, i.e. the extent to which the information system contains similarly structured functions.

1.2. Test strategy

During development or maintenance, quality requirements will have been specified for the information system. The test activities must determine the extent to which these requirements have been satisfied. In liaison with the client, the system and/or subsystem quality characteristics to be tested are identified and their relative importance determined. The importance of each characteristic influences the thoroughness of the related test activities. The more important a quality characteristic, the more exacting and thorough the tests have to be and the greater the volume of work. The importance of the various characteristics should be determined in consultation with the client when the test strategy is being formulated; the information can then be used as TPA input. In the course of the TPA process, the volume of testing work is calculated on the basis of the test strategy.

While certain general quality requirements apply to the information system as a whole, there are also differences between the various functions in terms of the requirements to be met. From the user's point of view, a function which is utilized throughout the day may be much more important than a processing function which operates only at night. For each function, therefore, there are two (subjective) factors that influences the thoroughness of testing namely the user-importance of the function and the usage-intensity. The thoroughness of testing reflects the degree of certainty or insight into system quality sought by the customer. The user-importance and usage-intensity factors are, of course, based on the test strategy.

As previously indicated, the importance attached to the various quality characteristics for testing purposes and the importance of the various subsystems and/or functions determine the test strategy. The test strategy specifies which quality characteristics are to be tested for each subsystem or function, and the relevant degree of coverage. TPA and strategy determination are closely related and in practice are often performed at the same time.

1.3. Productivity

Productivity is not a new concept to anyone who has produced estimates on the basis of function points. In function point analysis, productivity is an expression of the relationship between the number of hours necessary for a task and the measured number of function points. In TPA, productivity relates to the time necessary to realize one test point, as determined by the size of the information system and the test strategy. Productivity has two components: a productivity figure and an environmental factor. The productivity figure is based on the knowledge and skill of the test team, and is therefore specific to the individual organization. The environmental factor indicates the degree to which the environment influences the test activities to which the productivity is related. Influential environmental considerations include the availability of test tools, the amount of experience the team has with the test environment, the quality of the test basis and the availability of testware.

2. Basic procedure

The TPA procedure is illustrated below. The number of test points necessary for testing the dynamic measurable quality characteristics is calculated for each function on the basis of the number of function points assigned to the function, the function-dependent factors (complexity, interfacing, uniformity, user-importance and usage-intensity) and the quality requirements regarding or test strategy for the dynamic quality characteristics. The sum of the test points assigned to the individual functions is the number of “dynamic test points”. The number of test points necessary for testing the static measurable quality characteristics is calculated on the basis of the total number of function points for the information system and the quality requirements or test strategy for the static quality characteristics. This gives the number of static test points.

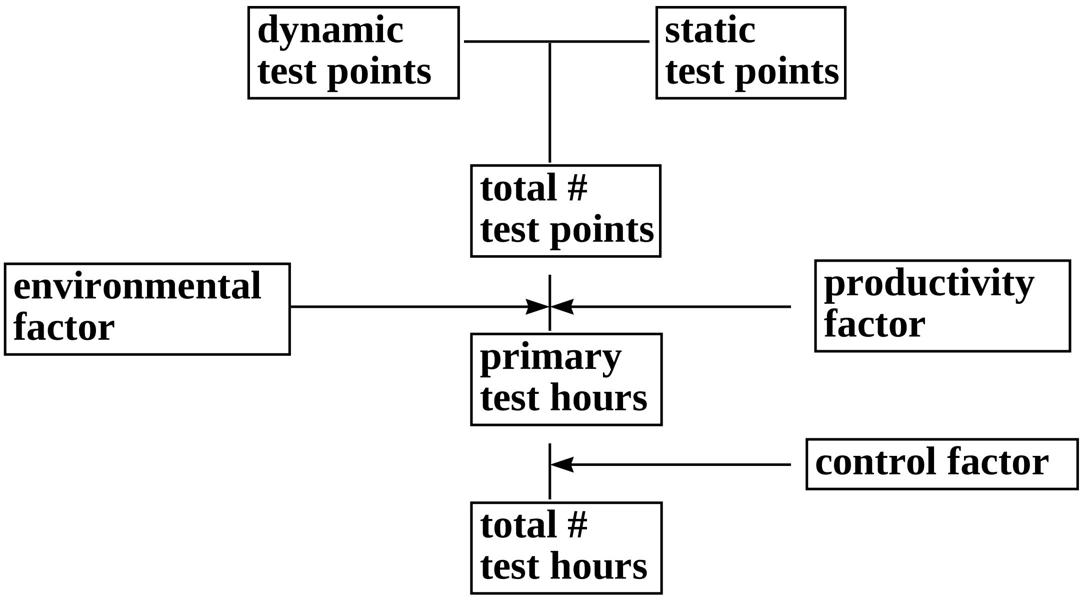

Figure 1: Overview Test Point Analysis procedure

The total number of test point is the sum of the dynamic and static test points. The number of primary test hours can then be calculated by multiplying the total number of test points by the calculated environmental factor and the applicable productivity factor. The primary test hours represents the volume of work involved in the primary testing activities, i.e. the time required for completion of the test phases Preparation, Specification, Execution and Completion [5], [6].

Finally, the total number of test hours is obtained by adding an allowance for secondary test activities (Planning and Control) to the primary number of test hours. The size of this allowance, which represents the volume of work involved in the management activities, depends on the size of the test team and the availability of management tools. The total number of test hours is an estimate of the time required for all test activities, excluding formulation of the test plan.

3. Principles

In TPA, the following principles apply:

- TPA is concerned only with the measurable quality characteristics (according to ISO 9126 [3]) which fall within the scope of acceptance testing and/or system testing. A characteristic is considered "measurable" if an appropriate test technique is available. Sufficient practical experience should already have been acquired using the test technique for the quality characteristic in question to allow the volume of work to be predicted with reliability.
- The corollary of the first principle is that, using the TPA technique in its present form, it is not possible to allow for all the quality characteristics that might be addressed by acceptance testing and/or system testing. The characteristics for which allowance cannot be made are those for which no pre-defined test technique (yet) exists and those in relation to which insufficient practical experience has been acquired. It is likely that any subsequent version of the TPA system would cover more quality characteristics.
- TPA is in principle analyst-independent. In other words, if two different people performed a TPA of the same information system, they should obtain the same result. This is because clear

rating definitions are provided for all factors which can be rated on an objective basis, while all other factors are determined by the customer.

- TPA depends on the availability of a function point count produced using Nefpug, Ifpug or IFPA. If Nefpug or Ifpug has been used, the gross function point count is used as the basis for analysis.
- For TPA purposes, the test team's matter knowledge is not treated as a variable which affects the amount of work involved in the tests. Naturally, it is assumed that the team does have a certain amount of matter knowledge. Sufficient matter knowledge is thus a precondition, satisfaction of which has to be assured at the test planning stage.
- TPA estimates are made on the assumption that on average one complete re-test will be conducted. This is a weighted average based on the size of the functions, expressed in test points.

4. TPA, the technique in detail

4.1. Input and starting conditions

To conduct a TPA, one needs access to a functional design, consisting of detailed process descriptions and a logical data model, preferably including a CRUD table. In addition, a function point count made using the Nefpug [7], Ifpug [2] or IFPA [8] technique is necessary. A count made using one of these three techniques can be used as input for the TPA. However, when determining a productivity factor from the historical data, it is important that only one of these function point-counting methods is used; different methods should not be combined. If Nefpug or Ifpug is used, the gross function point count is used as the basis for analysis. The choice of function point counting technique does not affect test point calculation, but it can influence the productivity factor. For TPA purposes, the function point count is amended as follows:

- The function points for the various (logical) data sets defined within the function point count are assigned to the function(s) which provide(s) the input for those (logical) data sets.
- The function points for the various linked data sets defined within the function point count are assigned to the function(s) which use(s) those linked data sets. This does not apply, of course, where counts made using IFPA are concerned, since this system does not recognize linked data sets.
- The number of function points for a clone-class FPA function is the same as the number of points assigned to the relevant original FPA function. A clone is an FPA function which has already been specified and/or realized within the same or another user function in the same project.
- The number of function points for a dummy-class FPA function is if possible calculated; otherwise, such functions are treated as being of average complexity and the corresponding number of function points are assigned. A dummy is an FPA function whose functionality does not need to be specified and/or realized, but which is nevertheless available because specification and/or realization has been undertaken outside the project.

If no function point count is available, but one is considered desirable (for TPA purposes), the time needed to carry out the count can be determined as follows:

The number of logical data sets is counted and multiplied by thirty. This gives a very rough approximation of the function point count. This approximation is divided by four hundred to obtain the number of days necessary for the count. (note: it is generally possible to count four hundred to five hundred function points a day.)

4.2. Dynamic test points

The number of dynamic test points is the sum of the test points assigned to the individual functions. To calculate the test points for the individual functions, the influential variables and factors are divided into two categories:

- Function-dependent (D_f)
- Quality requirements relating to the dynamic quality characteristics to be tested (Q_d)

The FPA function is used as the unit of function. The calculation of user-importance and usage-intensity is based largely on the user function as communication medium. The importance which users accord to a user function applies to all subsidiary FPA functions as well.

Function-dependent factor (D_f)

The various function dependent factors and the associated ratings are described below. One of the ratings given must be selected; intermediate ratings are not allowed. If insufficient information is available to enable rating of a given factor, the nominal rating (printed bold) should be assigned.

User-importance

The user-significance is an expression of the importance that the user attaches to a given function relative to the other system functions. A useful rule of thumb is that about 25 per cent of functions should be placed in the “high” category, 50 per cent in the “normal” category and 25 per cent in the “low” category.

User-importance is assigned to the functionality identified by the user. This means assigning user-importance to the user function. The user-importance of a function should, of course, be determined in liaison with the sponsor and other representatives of the user organization.

Rating:

3	Low: the importance of the function relative to the other functions is low.
6	Normal: the importance of the function relative to the other functions is normal.
12	High: the importance of the function relative to the other functions is high.

Usage-intensity

The usage intensity has been defined as the frequency with which a certain function is processed by the users and the size of the user group that uses the function. As with user-importance the usage-intensity is being determined at a user-function level.

Rating:

2	Low: the function is only used a few times per day or per week.
4	Normal: the function is being used a great many times per day
12	High: the function is used continuously throughout the day.

Interfacing

Interfacing is an expression of the extent to which a modification in a given function affects other parts of the system. The degree of interfacing is determined by ascertaining first the logical data sets (LDSs) which the function in question can modify, then the other functions which access these LDSs.

An interface rating is assigned to the function by reference to a table in which the numbers of LDSs affected by the function are ranged vertically and the numbers of other functions accessing the LDSs are ranged horizontally. When working out the number of "other functions" affected, a given function may be counted several times over if it accesses several LDSs, all of which are maintained by the function for which the interweave calculation is being made.

LDS\functions	1	2-5	>5
1	L	L	A
2-5	L	A	H
>5	A	H	H

Table 1 : Complexity table interface factor

Explanation : L : Low interfacing, A : Average interfacing ,H : High interfacing

If a function does not modify any LDSs, it is given a low interface rating. A CRUD table is very useful for determining the degree of interfacing.

Rating:

2	The degree of interfacing associated with the function is low.
4	The degree of interfacing associated with the function is normal.
8	The degree of interfacing associated with the function is high.

Complexity

The complexity of a function is determined on the basis of its algorithm. The general structure of the algorithm may be described using pseudo code, Nassi-Shneiderman or ordinary text. The complexity rating of the function depends on the number of conditions in the function's algorithm. When counting the conditions, only the processing algorithm should be considered. Conditions which are the result of database checks, such as domain validations or physical presence checks do not count, since these are implicitly included in the function point count.

Composite conditions such as "IF a AND b, THEN" count double, since, without the "AND" statement, two "IF" statements would be needed. Similarly, a "CASE" statement with n cases counts as n-1 conditions, since replacement of the "CASE" statement with a series of "IF" statements would result in n-1 conditions. To summarize: count the (simple) conditions, not the operators.

Rating:

3	The function contains no more than five conditions.
6	The function contains between six and eleven conditions.
12	The function contains more than eleven conditions.

Uniformity

Under the following circumstances, only 60% of the test points assigned to the function under analysis count towards the system total:

- In the case of a second occurrence of a virtually unique function: in such cases, the test specifications can be largely reused.
- In the case of a clone function: the test specifications can be reused for clone functions.
- In the case of a dummy function (provided that reusable test specifications have already been drawn up for the dummy).

A uniformity factor of 0.6 is assigned in cases of the kinds described above; otherwise a uniformity factor of 1 is assigned. An information system may therefore contain functions that possess a degree of uniformity for test purposes, even though they are regarded as unique in the context of a function point analysis. In function point analysis, the term “unique” is applied to the following:

- A function which uses a combination of data sets which is not used by any other input function
- A function which, although it does not use a unique combination of data sets, does use a unique processing technique (e.g. a unique method of updating a data set)

Conversely, an information system may contain functions which, although they are regarded as completely uniform in the context of a function point analysis and therefore do not warrant any function points, do count in TPA, since they do need to be tested. Clones and dummies come under this heading.

Method of calculation

The D_f factor is calculated by adding together the ratings for the first four function-dependent variables (user-importance, usage-intensity, interfacing and complexity) and dividing the sum by twenty (the nominal rating). The result should then be multiplied by the uniformity factor. A D_f factor is calculated for each function.

D_f = ((Ue + Uy + I + C)/16) * U

D_f	= weighting factor for the function-dependent factors
Ue	= user-importance
Uy	= usage-intensity
I	= interfacing
C	= complexity
U	= uniformity

Standard functions

If, as is often the case, the function point count includes the error report function, help-screen function and/or menu structure function, standard numbers of test points can be assigned, as indicated in the table below.

Function	FP's	Ue	Uy	I	C	U	A_f
Error message	4	6	8	4	3	1	1,05
Help screens	4	6	8	4	3	1	1,05
Menus	4	6	8	4	3	1	1,05

Table 2: Test points standard functions

Dynamic quality characteristics (Q_d)

The paragraphs below describe how the requirements relating to dynamic measurable quality characteristics are taken into account in the TPA process. In this context, distinction is made between implicitly and explicitly measurable quality characteristics.

In TPA, four dynamic, explicitly measurable quality characteristics are recognized:

- Suitability
- Security
- Usability (regarding usability no distinction has (yet) been made in subcharacteristics, since there are no usability testing techniques available that have this level of accuracy.)
- Efficiency (for the same reason as mentioned at usability, efficiency is not split up into time-behaviour and resource-utilisation)

The importance of the requirements relating to each quality characteristic is rated; if necessary, this is done separately for each subsystem.

<u>Rating:</u>

0	Quality requirements are not important and are therefore disregarded for test purposes.
3	Quality requirements are relatively unimportant but do need to be taken into consideration for test purposes.
4	Quality requirements are of normal importance. (This rating is generally appropriate where the information system relates to a support process.)
5	Quality requirements are very important. (This rating is generally appropriate where the information system relates to a primary process.)
6	Quality requirements are extremely important.

Dynamic, explicitly measurable quality characteristics:

Characteristic / Rating:	0	3	**4**	5	6
Functionality	(weighting 0.75)				
Security	(weighting 0.05)				
Usability	(weighting 0.10)				
Efficiency	(weighting 0.10)				

Where black-box tests are concerned, the elementary comparative test (ECT), data flow test (DFT), semantic test and syntactic test are available for testing suitability; the semantic test (SEM) is available for testing security; the process cycle test (PCT), use cases and SUMI are available for testing usability and the real-life test (RLT) is available for testing efficiency (time-behaviour and resource-utilisation) For more information on the various testing techniques see [4] and [5].

The table below illustrates how the choice of test specification techniques is often related to the rating given to the dynamic quality characteristics.

Rating :	3	**4**	5	6
Suitability • Processing : • Screen checks :	DFT and Error Guessing Error guessing	DFT sample SEM and Error guessing	EVT and DFT sample SEM and SYN	EVT SEM and sample SYN
Security	Error Guessing	SEM sample user profiles	SEM user profiles	SEM user profiles and overall system *
Usability	No testspec's and SUMI	Use Case or PCT (Ue : high) and SUMI	Use Case or PCT (Ue : average, high) and SUMI	Usability laboratory test
Efficiency	The *thoroughness* of the RLT is variable and will thus be determined by the rating and the amount of hours that comes available as a consequence.			

Table 3: Test techniques versus quality characteristics

* If the security characteristic is given a rating of six, the semantic test should be used to examine the user profiles and associated access privileges both for the information system to be tested and for the infrastructure or information network as a whole.

One has to determine which quality characteristics will be tested dynamic implicitly. A statement regarding these quality characteristic can be done by gathering data during test execution. ISO 9126 part 2 “External Metrics” can be used as source of information or inspiration for this [4]. For instance Time-behaviour can be explicitly tested applying the Real-life test or implicitly by gathering data and establishing metrics. The dynamic implicit quality characteristics need to be specified. Subsequently the number of quality characteristics can be determined to which external metrics will be applied. Per characteristic a rating of 0.02 is applicable in the context of Q_d.

Method of calculation (Q_d)

The rating for each dynamic, explicitly measurable quality characteristic is divided by four (the nominal rating), then multiplied by the weighting factor. Next, the ratings thus calculated for the five dynamic, explicitly measurable quality characteristics are added together.

If certain dynamic, implicitly measurable quality characteristics are to be included in the test, the appropriate weighting (0.02 for each characteristic) should be added to the result obtained for the dynamic, explicitly measurable quality characteristics. The figure thus calculated is the Q_d factor. Normally, a single Q_d factor can be calculated for the system as a whole. However, if different test strategies are to be used for the various subsystems, a separate Q_d factor calculation should be made for each subsystem.

Dynamic test point formula

The number of direct test points is the sum of the test points assigned to the individual functions. The number of test points assigned to each function can be calculated by entering the data so far obtained into the following formula:

$$\mathbf{TP_f = FP_f * D_f * Q_d}$$

TP_f	= number of test points assigned to the function
FP_f	= number of function points assigned to the function
D_f	= weighting factor for the function-dependent factors
Q_d	= weighting factor for the dynamic quality characteristics

4.3. Static test points

The indirect test point count depends mainly on the function point count for the system as a whole. The indirect test point count is also influenced by the requirements regarding the static quality characteristics to be tested (the Q_i factor).

One has to determine whether the static measurable quality characteristics are relevant for test purposes. A static test can be carried out using a checklist. In principle all ISO 9126 quality characteristics [3] can de tested using a checklist.

E.g. Security can therefore be measured dynamically, using a semantic test, and/or statically, by evaluating the security measures with the support of a checklist.

Method of calculation (Q_i)

If a quality characteristic is tested by means of a checklist (static test), the factor Q_i get the value sixteen. For each subsequent quality characteristic to be included in the static test, another sixteen is added to the Q_i factor rating.

4.4. Total number of test points

The total number of test points assigned to the system as a whole is calculated by entering the data so far obtained into the following formula:

$$\mathbf{TP = \Sigma TP_f + (FP * Q_i) / 500}$$

TP	= total number of test points assigned to the system as a whole
ΣTP_f	= sum of the test points assigned to the individual functions (dynamic test points)
FP	= total number of function points assigned to system as a whole (minimum value 500)
Q_i	= weighting factor for the indirectly measurable quality characteristics

4.5. Primary test hours

The formula presented in subsection 5.4 gives the total number of test points assigned to the system as a whole. This total number of test points is a measure of the volume of the primary test activities. The primary number of test points are multiplied by the productivity factor and the environmental factor to obtain the primary test hour count. The primary test hour count is the number of hours required for carrying out the test activities involved in the test life cycle phases Preparation, Specification, Execution and Completion.

Productivity factor

The productivity factor indicates the number of test hours required per test point. The higher the productivity factor, the greater the number of test hours required. The productivity factor is a measure of the experience, knowledge and skill of the test team. The productivity factor can vary from one organization to the next or from one organizational unit to the next. Productivity factors can be calculated by analyzing completed test projects; thus, historical data on such projects is necessary for productivity factor determination.

In practice the productivity factor has shown to have a value between 0,7 and 2.0.

Environmental factor

The number of test hours required for each test point is influenced not only by the productivity factor, but also by the environmental factor. A number of environmental variables are defined for calculation of the environmental factor. The various environmental variables and the associated ratings are described below. Again, one of the ratings given must be selected; intermediate ratings are not allowed. If insufficient information is available to enable rating of a given variable, the nominal rating (printed bold) should be assigned.

Test tools

The test tools variable reflects the extent to which testing is automated, or the extent to which automatic tools are used for testing. For the purpose of calculating this variable, the term "test tools" covers tools that are used for the primary test activities. The availability of test tools means that some of these activities can be performed automatically and therefore more quickly.

<u>Rating:</u>

1 Testing involves the use of a query language such as SQL; a record and playback tool is also being used.

2 Testing involves the use of a query language such as SQL, but no record and playback tool is being used.

4 No test tools are available.

Development testing

The development testing variable reflects the quality of earlier testing. If the estimate under preparation is for an acceptance test, the earlier testing will have been system testing; if the estimate is for a system test, the earlier testing will have been white-box testing. The quality of such development testing influences the amount of functionality that *may* be require less thorough testing with less coverage and the duration of the test activities. For, the better the development testing, the less likely one is to encounter time-consuming problems during the test currently under consideration.

<u>Rating:</u>

2 A development testing plan is available and the test team is familiar with the actual test cases and test results

4 A development testing plan is available.

8 No development testing plan is available.

Test basis

The test basis variable reflects the quality of the (system) documentation upon which the test under consideration is to be based. The quality of the test basis influences the amount of time required for the Preparation and Specification phases.

<u>Rating:</u>

3 During the system development documentation standards are being used and a template, in addition the inspections are organized

6 During the system development documentation standards are being used and a template.

12 The system documentation was *not* developed using a specific standards and a template.

Development environment

The development environment variable reflects the nature of the environment within which the information system was realized. In this context, the degree to which the development environment will have prevented errors and inappropriate working methods is of particular importance. If errors of given type cannot be made, it is of course not necessary to test for them.

<u>Rating:</u>

2 The system was developed using a 4 GL programming language with an integrated DBMS containing numerous constraints.

4 The system was developed using a 4 GL programming language, possibly in combination with a 3 GL programming language.

8 The system was developed using only a 3 GL programming language such as COBOL, PASCAL or RPG.

Test environment

The test environment variable reflects the extent to which the test infrastructure in which the testing is to take place has previously been tried out. In a well-tried test infrastructure, fewer problems and delays are likely during the execution phase.

Rating:

1 The environment has been used for testing several times in the past.

2 The test is to be conducted in a newly equipped environment similar to other well-used environments within the organization.

4 The test is to be conducted in a newly equipped environment which may be considered experimental within the organization.

Testware

The testware variable reflects the extent to which the tests can be conducted using existing testware. The availability of usable testware mainly influences the time required for the Specification phase.

Rating:

1 A usable general initial data set (tables, etc.) and specified test cases are available for the test.

2 A usable general initial data set (tables, etc.) is available for the test.

4 No usable testware is available.

Method of calculation (E)

The environmental factor (E) is calculated by adding together the ratings for the various environmental variables (test tools, development testing, test basis, development environment, test environment and testware), then dividing the sum by twenty-one (the sum of the nominal ratings). Normally, one environmental factor is worked out for the system as a whole, but separate factors can be calculated for the individual subsystems if appropriate.

Primary test hours formula

The number of primary test hours is obtained by multiplying the number of test points by the productivity factor and the environmental factor:

$$\mathbf{PT = TP * P * E}$$

PT	= the total number of primary test hours
TP	= the total number of test points assigned to the system as a whole
P	= the productivity factor
E	= the environmental factor

4.6. Total number of test hours

Since every test process involves tasks which may be placed under the heading “planning and control”, allowance needs to be made for such activities. The number of primary test hour and the planning and control allowance together give the total number of test hour.

The standard (nominal) allowance is 10 per cent. However, the allowance may be increased or decreased, in line with the following two factors:

- Team size
- Management tools

Team size

The team size factor reflects the number of people making up the team (including the test manager and, where appropriate, the test controller).

<u>Rating:</u>

3	The team consists of no more than four people.
6	The team consists of between five and ten people.
12	The team consists of more than ten people.

Planning and control tools

The planning and control tools variable reflects the extent to which automated resources are to be used for planning and control.

<u>Rating:</u>

2	Both an automated time registration system and an automated defect tracking system (including CM) are available.
4	Either an automated time registration system or an automated defect tracking system (including CM) is available.
8	No automated (management) systems are available.

Method of calculation

The planning and management percentage is obtained by adding together the ratings for the two influential factors (team size and planning and control tools). The allowance in hours is calculated by multiplying the primary test hour count by this percentage. Addition of the planning and control allowance to the number of primary test hours gives the total number of test hours.

4.7. Breakdown between phases

The result of a TPA is an estimate for the complete test process, excluding formulation of the test plan. If a structured testing approach [5],[6] is used, the test process is divided into five life cycle phases; many clients will want to see estimates for the individual phases, as well as for the complete test process. The estimate for the Planning and Control phase will normally be the same as the planning and control allowance, i.e. the primary test hour count multiplied by the planning and control percentage. The primary test hours are then divided between the Preparation, Specification, Execution and Completion phases. The breakdown between the phases can of course vary from one organization to another, or even from one organizational unit to another. Suitable phase percentages can be calculated by analyzing completed test projects; thus, historical data on such projects is necessary for breaking down the total estimate.

Experience with the TPA technique suggests that the following percentages are generally appropriate:

• Preparation:	10 percent
• Specification	40 percent
• Execution	45 percent
• Completion	5 percent

5. TPA at an early stage

A test project estimate is often needed at an early stage. Until detailed functional specifications are obtained, however, it is not possible to determine factors such as complexity, interfacing and the like. Nevertheless, a rough function point analysis can often be performed on the basis of very general specifications. If a rough function point count is available, a rough TPA can be performed as well.

For a rough TPA, a single function is defined whose size is determined by the total (gross) function point count. All function-dependent factors (user-importance, user-intensity, complexity, interfacing and uniformity) are usually assigned a normal value, so that D_f has a value of one. A TPA can then be carried out as described in section 5. The environmental factor will often have to be based on assumptions; any such assumptions should be clearly documented and stated on the test estimate when it is presented to the client.

6. References

[1] Albrecht, A.J. (1984), AD/M productivity measurement and estimate validation, IBM Guideline

[2] IFPUG (International Function Point User Group) (1994), Function Point Counting Practices, release 4.0, IFPUG, January 1994

[3] ISO/IEC FCD 9126-1 (1998), Information technology - Software product quality – Part 1: Quality Model, International Organization of Standardization

[4] ISO/IEC PDTR 9126-2 (1997), Information technology - Software product quality - Part 2: External metrics, International Organization of Standardization

[5] Pol, M., R. Teunissen and E. van Veenendaal (1995), Testing according to Tmap (in Dutch), Tutein Noltenius, ‘s Hertogenbosch, The Netherlands

[6] Pol, M and E. van Veenendaal (1999), Structured Testing; an introduction to TMap, Kluwer Bedrijfsinfomatie, Deventer, The Netherlands

[7] NEFPUG (Dutch Function Point User Group), (1991), Definitions and counting guidelines for the appliaction of function point analysis (in Dutch), NEFPUG, Amsterdam, May 1991

[8] Schimmel, H.P. (ed.) (1989), InterprogramFunctionPointAnalysis (IFPA) (in Dutch) , Samson Publishing, Alphen aan den Rijn, The Netherlands

[9] Veenendaal, E.P.W.M. van (1995), Test Point Analysis: a method for estimating the testing effort (in Dutch), in: Computable, May 1995