Test Effort Estimation with Test Point Analysis

Ruud Teunissen
Test effort estimation with Test Point Analysis

Ruud Teunissen
Gitek nv - *interaction through software*

EuroSTAR 2000
Topics

- Test effort estimation
- Systematic software testing
- Risk based testing
- Test effort estimation with Test Point Analysis
- Conclusion
Test effort estimation

- Possible approaches:
  - top down according to permitted budget
  - extrapolation (expected) design- and/or build time
  - related to functionality (FP’s, #pages FD)
  - bottom up from work breakdown
  - intuition of experienced testers
  - based on comparison with earlier test projects
  - test point analysis (TPA®)
  - ........
Required: *Structure*

**TEST** → **DEFECTS, ADVICE**

- Informing about quality & risks
- Delivering re-usable testware
- Preventing defects

**I** - Infrastructure
**L** - Life-cycle
**O** - Organisation
**T** - Techniques

- What, when?
- How?
- Where, etc.?
- Who?
TMap® testing life cycle

Preparation

Specification

Execution

Completion

Planning & Control

P
S
E
C

P&C

Life Cycle
Test Strategy

Aim: Find the most important defects as early as possible at the lowest price

- Risks (business, project, test)
- Quality characteristics
- Available resources
  - people & expertise
  - infrastructure and tools
  - techniques
- Budget and time

What are the most important defects?
Risk thinking: Introduction

- Business reasons
- No risk, no test
- Risk analysis approach
Risk: Definition

Risk

Chance of failure

Frequency of use

Chance of error

Damage

interaction through software
Parties involved

Frequency of use
Damage

Chance of error

TEST

- Project management
- Accountancy
- Functional management
- System management
- Business
- Data center
- DBA
- CM & CC
- QA
- Technicians
- Design
- Programmers
# Test importance per Q-char/subsystem

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>s1 %</th>
<th>s2 30</th>
<th>s2 15</th>
<th>s3 20</th>
<th>conv 15</th>
<th>tot. 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionality</td>
<td>60</td>
<td>+++</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>security</td>
<td>5</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>time behavior</td>
<td>5</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>suitability</td>
<td>20</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>usability</td>
<td>10</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>
Strategy based effort estimation

Test Point Analysis

- Size
- Strategy
- Productivity

Test effort
TPA, The model

Dynamic Testing

Test-Strategy

Static Testing

Total Test points

Environmental factor

Productivity factor

Primary Test hours

Management overhead

Total Test hours
Dynamic and static testing

- **Dynamic testing**
  - activities related to individual functions or subsystems
  - explicit testing (dedicated test cases)
  - implicit testing (evaluate certain characteristics during explicit tests)

- **Static testing**
  - activities related to the system as a whole
  - verification (checklists)
Dynamic Testing

- Dynamic test points = \( \sum (\text{FP} \times \text{T} \times \text{Qd}) \)
  - FP = Size in Function Points per function
  - T = Test impact per function
  - Qd = Dynamic quality attributes
Function Point Analysis

- Establish size of functionality based on:
  - number of functions:
    - number of screens and fields
    - number of reports and fields
    - processing functions
  - number of files and fields
- Measurement unit: function point (FP)
- Accepted standard, worldwide
- Productivity (=hours/fp) depending on organisation and programming language
Test impact per function

- user-importance
- usage-intensity
- interfacing
- complexity
- uniformity
User-Importance

- The user-importance is an expression of the importance that the user attaches to a given function relative to the other system functions.

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

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

Rating

2 Low: the function is executed by the user organisation only a few times per day or per week;

4 Normal: the function is being executed by the user organisation a great many times per day;

8 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.

<table>
<thead>
<tr>
<th>LDS\functions</th>
<th>1</th>
<th>2-5</th>
<th>&gt;5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L</td>
<td>L</td>
<td>A</td>
</tr>
<tr>
<td>2-5</td>
<td>L</td>
<td>A</td>
<td>H</td>
</tr>
<tr>
<td>&gt;5</td>
<td>A</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>Factor</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>user-importance</td>
<td>3</td>
</tr>
<tr>
<td>usage-intensity</td>
<td>2</td>
</tr>
<tr>
<td>interfacing</td>
<td>2</td>
</tr>
<tr>
<td>complexity</td>
<td>3</td>
</tr>
<tr>
<td>uniformity</td>
<td>0.6</td>
</tr>
</tbody>
</table>

\[ T(f) = \left( \frac{3 + 4 + 4 + 12}{20} \right) * 1 = 1.15 \]
Standard Functions

- error report function
- help-screen function
- menu structure function

<table>
<thead>
<tr>
<th>Function</th>
<th>FP's</th>
<th>Ue</th>
<th>Uy</th>
<th>I</th>
<th>C</th>
<th>U</th>
<th>Df</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>Help screens</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>Menus</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1.05</td>
</tr>
</tbody>
</table>
Dynamic Testing

- Dynamic test points = \( \sum (FP \times T \times Qd) \)
  - FP = Size in Function Points per function
  - T = Test impact per function
  - Qd = Dynamic quality attributes
Dynamic quality attributes (explicit)

- Required depth (test coverage) of explicit tests

Rating

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 average importance (this rating is generally appropriate when the information system relates to a support process);

5  Quality requirements are very important (this rating is generally appropriate when the information system relates to a primary process);

6  Quality requirements are extremely important.
Dynamic quality attributes

- Balanced, using a weight factor
  - functionality (0.75) 0 3 4 5 6
  - security (0.05) 0 3 4 5 6
  - suitability (0.10) 0 3 4 5 6
  - performance (0.10) 0 3 4 5 6

\[
Q_{d'} = \left( \frac{5}{4} \right) \times 0.75 + \left( \frac{3}{4} \right) \times 0.05 + \left( \frac{5}{4} \right) \times 0.10 + \left( \frac{4}{4} \right) \times 0.10 = 1.20
\]
Dynamic quality attributes (implicit)

- User friendliness (0.02) \( Y \quad N \)
- Resource usage (0.02) \( Y \quad N \)
- Performance (0.02) \( Y \quad N \)
- Maintainability (0.04) \( Y \quad N \)

\[
Q_d(implicit) = 0.02 + 0 + 0 + 0.04 = 0.06
\]

\[
Q_d = 1.20 + 0.06 = 1.26
\]
### Rating versus test design techniques

<table>
<thead>
<tr>
<th>Rating</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Suitability</strong>&lt;br&gt;• Processing</td>
<td>DFT and Error Guessing</td>
<td>DFT</td>
<td>ECT and DFT</td>
<td>ECT</td>
</tr>
<tr>
<td>• Screen checks</td>
<td>Error guessing</td>
<td>sample SEM and Error guessing</td>
<td>sample SEM and SYN</td>
<td>SEM and sample SYN</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Error Guessing</td>
<td>SEM sample user profiles</td>
<td>SEM user profiles</td>
<td>SEM user profiles and overall system</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>No tests specs and SUMI</td>
<td>Use Cases or PCT and SUMI</td>
<td>Use Cases or PCT and SUMI</td>
<td>Usability Laboratory Test</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DFT = Data Flow Test<br>ECT = Elementary Comparison Test<br>SEM = Semantic Test<br>Syn = Syntactic Test<br>PCT = Process Cycle Test<br>SUMI = Software Usability Measurement Inventory<br>RLT = Real Life Test
Static testing

- \( TP(\text{ indirect }) = Qs \times \frac{FP}{500} \)
  - \( Qs \) = Static quality attributes to be tested
  - \( FP \) = Total amount of function points of the system

Each static quality attribute costs 16 TP’s / 500 FP’s
Example: static test points

- Size of the system = 3500 FP
- Static quality attributes
  - flexibility Y N
  - testability Y N
  - security Y N
  - continuity Y N
  - traceability Y N

\[ TP( \text{indirect} ) = (16 + 16) \times 3500 / 500 = 224 \]
Environmental factor

- Test tool support
- Development tests
- Product documentation
- Development environment
- Test environment
- Testware
Test Tools

- The test tools variable reflects the extent to which the testing activities are being supported by automatic tools.

Rating
1. During testing a supporting tool for test specification is applied and a tool is used for 'record & playback';
2. During testing a supporting tool for test specification is applied or a tool is used for 'record & playback';
3. No test tools are available.
Development Testing

- The development testing variable reflects the quality of earlier testing.

Rating

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.

Rating

3  During system development documentation standards and templates are being used. In addition to this, reviews are organized.

6  During system development documentation standards and templates are being used, but their use is not checked in any formal way;

12 The system documentation was not developed using specific standards and templates.
### Development Environment

- The development environment variable reflects the nature of the environment within which the information system was realized.

#### Rating

<table>
<thead>
<tr>
<th>Rating</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>The system was developed using a 4 GL programming language with an integrated DBMS containing numerous constraints;</td>
</tr>
<tr>
<td>4</td>
<td>The system was developed using a 4 GL programming language, possibly in combination with a 3 GL programming language;</td>
</tr>
<tr>
<td>8</td>
<td>The system was developed using only a 3 GL programming language</td>
</tr>
</tbody>
</table>
Test Environment

- The test environment variable reflects the extent to which the test infrastructure in which testing is to take place, has previously been tried out.

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;
3. The test is to be conducted in a newly equipped environment, which may be considered experimental within the organization.
The testware variable reflects the extent to which the tests can be conducted using existing testware.

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.
### Example: factor conditions

<table>
<thead>
<tr>
<th>External condition</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test tool support</td>
<td>1 2</td>
</tr>
<tr>
<td>Development tests</td>
<td>2 4 8</td>
</tr>
<tr>
<td>Product documentation</td>
<td>3 6 12</td>
</tr>
<tr>
<td>Development environment</td>
<td>2 4 8</td>
</tr>
<tr>
<td>Test environment</td>
<td>1 2 4</td>
</tr>
<tr>
<td>Testware</td>
<td>1 2 4</td>
</tr>
</tbody>
</table>

Factor = \( \frac{(1 + 8 + 6 + 4 + 4 + 4)}{21} = 1.29 \)
Productivity factor

- Depending on
  - experienced personnel
  - culture
  - work procedures
- Typical range: 0.7 - 2.0
Management overhead

- Team size
  - larger -> more coordination
- Management tool support
  - time registration and reporting
  - problem registration and reporting
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).

Rating:

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.
Management Tools

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

Rating

2 Both an automated time registration system and an automated defect tracking system (incl. configuration management) are available and are applied;

4 Either an automated time registration system or an automated defect tracking system (incl. configuration management) is available and is applied;

8 No automated (management) systems are available.
Example: Management overhead

- Added directly as percentage
  - team size 3 6 12
  - mgt. support 2 4 8

Percentage = 6 + 8 = 14%
Distribution of the test effort

- Preparation: 10%
- Specification: 40%
- Execution: 45%
- Completion: 5%

Overhead: an additional 5-20%
From business risks to estimated test effort!
Literature on Test Point Analysis


Questions?

Ruud Teunissen
Gitek n.v. interaction through software
e-mail: rt@gitek.be
http://www.gitek.be
Gitek n.v. - *interaction through software*

- Founded in September 1986
- Number of employees: + 100
- IT Services: services on design & realization of applications, system management, network management, project assistance or complete project realization.
- Test Expertise: advice, implementing a structured test process, planning and management, test design & execution, training & coaching in testing, total test projects
- Customers in financial and insurance sector, public sector, pharmaceutical and telecommunications industry, chemical industry
Test Effort Estimation with Test Point Analysis
Ruud Teunissen

Since 1989 Ruud Teunissen is employed in the testing world. Ruud has performed several test functions in a number of ICT projects: tester, test specialist, test consultant, test manager. Ruud participated in the development of the structured testing methodology TMap® and is co-author of several books on structured testing.

As manager testen for Gitek n.v. in Belgium, Ruud is responsible for the testing department that employs 40 professional Test Engineers. Ruud is frequently speaking at conferences in Europe.

Books:

