# Sources of Error in Software Cost Estimation

Seminar on Software Cost Estimation
WS 02/03

Presented by
Silvio Meier
smeier@ifi.unizh.ch

Requirements Engineering Research Group
Department of Computer Science
University of Zurich, Switzerland

Prof. M. Glinz
Arun Mukhija

**Date: January, 7th 2003**

**Content**

## 1. Accuracy of Estimating Tools

A general problem in software cost estimation is, that historical cost data often contains a bias, so that this data can not directly be used for

- development of cost estimation tools, respectively for the calibration of such tools and
- comparing estimations against its real (historical) costs.

Often available tools predict costs which are between 50 to 100 percent higher than the cost data tracked by cost tracking systems. Mostly, the currently available tools predict higher costs and longer schedules than historical data, that would be indicated by the corresponding projects. To get reliable cost data for the development of cost estimation tools and for determining the accuracy of cost estimations, there has to be first determined which parts of historical costs in software projects cause the bias. After determination, the bias can be eliminated.

### 1.1 Sources of Inaccurate Historical Cost Data

Cost tracking systems are seldom optimised and designed for cost tracking of software projects. One problem with conventional cost tracking systems, when using them for the cost tracking of software projects, is that there are many tasks or activities or even whole phases which are omitted. One reason is that the chart of accounts often does not match a full software life cycle. The following lines show examples of omitted costs in software cost tracking [Jones98]:

- Early requirements phase
- Unpaid overtime
- Specialists work
- Technical work by users
- Project management
- Travel costs

In general, we can identify the following three problems when tracking historical cost data [Jones98]:

- Failure to include all activities, that were performed
- Failure to include all classes of workers
- Failure to include unpaid overtime

The most common reasons for above problems are:

- The cost tracking was initialised after the first work for the system was started, so the first activities are not tracked.
- Work which is done by non-programming personnel like database administrators, technical writers, etc. is not tracked
- Project management work is not tracked (especially second level work or higher management work)
- Technical work performed by users is not tracked.

Omissions in software cost tracking are not the only sources of error when collecting historical data. There are other observations which were made during different software process assessments [Jones98]:

- When projects are running low on funding, there is the tendency to charge time to other projects.
- Some personnel refuse to use cost tracking systems at all.
- Other personnel (electronic engineers, mechanical engineers) are involved in the software project (even coding), especially when a complex system is developed, but

they don't see their work as software project participants, so they don't charge their time in the project cost tracking tool.
- There is no cost-tracking system at all, i.e. no historical data is available for use during accuracy checks.

### 1.3 Elimination and Avoidance of the Bias in Cost Data

To use historical data for accuracy checking or calibrating estimation tools, it is needed to correct the historical cost data. This is a very difficult task which is mainly solved by doing interviews with project members and managers of the corresponding projects to filter the activities which are biased. The following rough values were investigated when doing such interviews. The values are set into relation with the total project effort [Jones98]:

- 5 – 10 % will be spent before the cost tracking system is set up.
- 15 –30 % of the work is done by workers which are not included in a typical cost-tracking system.
- The management effort will amount to between 10 to 20 % of the overall project. This effort is sometimes not tracked.
- The user participation in technical work is between 5 to 20 % and is almost never tracked.
- Unpaid overtime of exempt professionals and managers are in the range of 5 – 15%

It can be stated, that the accuracy of cost data also depends on the industry as presented in the following Table 1.1 (from [Jones98]):

**Table 1.1**: Patterns of Missing Software Cost Data by Industry [Jones98]

| **Software subindustry** | **Percentage of missing data** | **Most common omissions** |
|---|---|---|
| Military software | 10 | Unpaid overtime |
| Contracted or outsourced software | 10 | Unpaid overtime |
| Systems software | 12 | Unpaid overtime and documentation |
| Commercial software | 15 | Unpaid overtime, user activities, noncode tasks, specialists and project managers |
| End-user software | 75 | Everthing but coding |

The correction of the biased historical can be done with one or more of the following three ways. These methods are also used by software cost estimation tools developers to get accurate cost data:

- Excluding incomplete projects from the project portfolio
- Correcting wrong data and add missing data. Both kind of correctionscan be done by interviewing project members and other staff associated somehow to the project.
- Building activity-based cost estimating tools

A general suggestion is to avoid data which is not granular enough. It is useful to breakdown the project into activities and tasks! Data on the level of phases (requirements, design, coding, testing, etc.) is not useful for cost estimation.
Also there are activities which are spread over more than one phase (e.g. configuration control, integration, preparation of user manual, etc.), that means that cost data of single phase activities should not be used for estimating multiphase activities.

A possible list of all activities and the most common omissions can be found in Table 1.2. Only 5 from 25 activities are accurate enough to be used directly (i.e. without corrections) for estimation purposes. Using inaccurate cost data implies higher productivity rates and makes projects look cheaper than they really are.

Table 1.2 shows the minimal level of granularity to get enough accurate historical data to judge the accuracy of software estimation tools. Data on the level of projects is not useful. For fine tuning of cost estimation tools, two additional lower levels to the existing two levels are needed: tasks and subtasks. If we take as an example Unit Testing, the tasks would be: Preparing, Running, Repairing. With all possible subtasks of a software project on the most granular level of precision, the cost tracking exceeds 1000 data elements.

**Table 1.2:** Activities in a Software Project and Completeness of Data [Jones98]

| **Activities performed** | **Completeness of historical data** |
|---|---|
| Requirements | Missing or incomplete |
| Prototyping | Missing or incomplete |
| Architecture | Incomplete |
| Project planning | Incomplete |
| Initial analysis and design | Incomplete |
| Detail design | Incomplete |
| Coding | Complete |
| Reusable code acquisition | Missing or incomplete |
| Purchased package acquisition | Missing or incomplete |
| Code inspections | Missing or incomplete |
| Independent verfification and validation | Complete |
| Configuration management | Missing or incomplete |
| Integration | Missing or incomplete |
| User documentation | Missing or incomplete |
| Unit testing | Incomplete |
| Function testing | Incomplete |
| Integration testing | Incomplete |
| System testing | Incomplete |
| Field testing | Incomplete |
| Acceptance testing | Missing or incomplete |
| Independent testing | Complete |
| Quality assurance | Missing or incomplete |
| Installation and training | Missing or incomplete |
| Project management | Missing or incomplete |
| Total project resources, costs | Incomplete |

## 2. Judging the Accuracy of Software Cost Estimates

When judging the accuracy of tools, there are three general issues which come to mind:

- How good are the results of tools compared to historical cost data?
- How good are different competing software cost estimation tools when they are compared against each other?
- How good are manual estimations compared to tool estimations?

### First Issue

As mentioned before, tools are often more accurate than historical cost data [Jones98]. When historical data is accurate, estimates usually come within 5 – 10 % percent, sometimes closer to the real costs.

### Second Issue

There are many comparisons of tools, therefore have a look at [Jones98] where different studies are described, most of them are within a military context.

### Third Issue

A study which is mentioned by [Jones98] shows that only 4 of 50 manual estimates (see Figure 1.1) were within a range of plus or minus 5 percent of the (corrected) historical costs.

When the responsible people were asked for the reasons of so optimistic estimations, some of the captured answers [Jones98] were:

- I could not get approval for an accurate estimate, so I had to change it
- The project doubled in size after the requirements.
- Debugging and testing took longer than we thought.
- The new case tools we were using didn't work right and slowed us down.
- We didn't have any estimating tools available at the same time the estimate was needed.
- I lost some of my developers and had to find replacements.

The most problems will be mentioned in one of the follwing 12 error classes in Section 3.

Compared to manual estimates, automated ones are usually accurate or at least conservative for costs and schedules. In the above mentioned study, about 22 of 50 automated software cost estimations were within a range of plus or minus 5 percent of the historical cost. Only one estimate out of 50 was too optimistic (see figure 1.2). The direction of errors is reversed, i.e. more conservative estimates are made.
In general we can say, that it would be the aim to be within this plus minus 5 percent range or at least to be not too optimistic, therefore automated estimation should be preferred.

**Figure 1.1:** Accuracy of 50 Projects with Manual Estimation Techniques [Jones98]

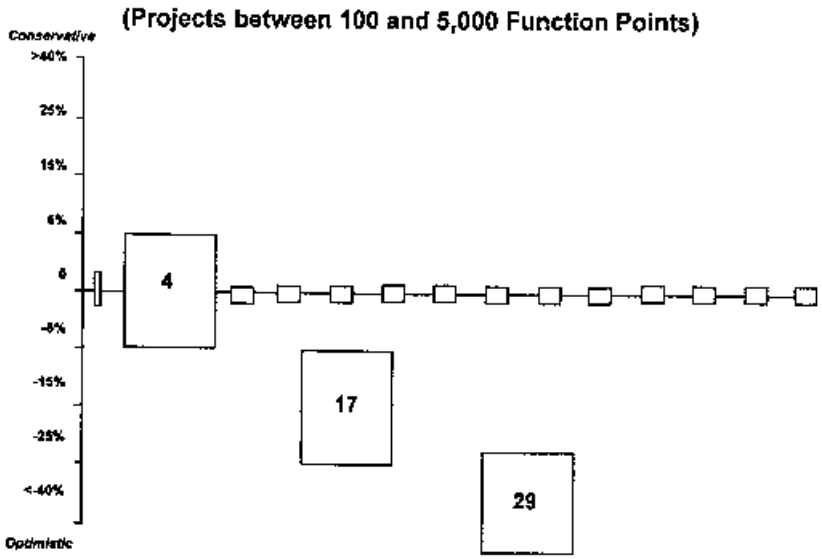
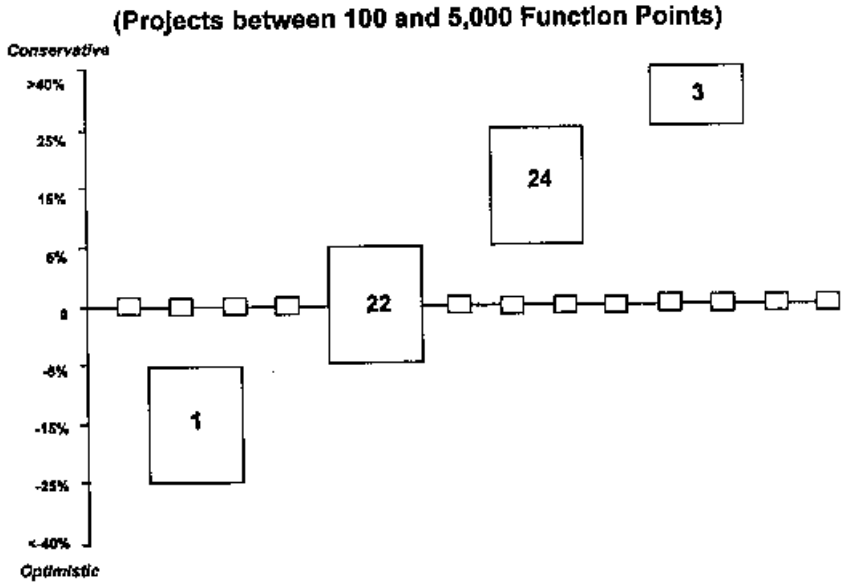


**Figure 1.2:** Accuracy of 50 Projects with Tool Estimation Techniques [Jones98]




## 3. Classes of Software Estimation Errors

The following error classes try to categorize all types of error being made by giving a description with an example of the error and a description of the error's impact on the cost estimation. In estimations often appear more than one of the following mentioned errors. The errors also influence each other or cause further errors, so the impact of these errors can result in a heavy error in the cost estimation.

### 3.1 Metrics Errors

**Error Description**

Doing software cost estimation by using LOC as metric is known to be unreliable since the 1970s.

There are two problems when using LOC as metric for estimating software costs:

- The way the code is counted influences the metric.
- LOC measures programs, but only the half of a software project effort is directly related to source code.

The latter is worse, i.e. the LOC metric tends to move in a wrong estimation direction when using it.

For example productivity measuring of higher level languages tends often to be estimated wrong, because higher level languages seem to have a lower productivity. This is because one can express with less LOC more functionality. Therefore, there is the paradoxical assumption that with higher level language increases the number of LOCs written in the same time as with a lower level language like assembler. But that's a wrong conclusion. Normally you will write less LOC with a higher level language than with a lower level language in the same time. For example, if a software developer writes 600 LOC of assembly language per month, it will be a false conclusion to assume that the productivity will be about 700 LOC per month, when using the more powerful C++ language.

The main problem of the LOC metric consists of two components:

1. The historical data based upon LOC is not granular enough to include all necessary activities.
2. The LOC metric is error-prone and dangerous for cross-language measurement or estimation purposes.

In contrast, the function point metric is not error-prone like the LOC metric, because function points can use historical data independent of language. The last few years, function points spread very rapidly because of these advantages.

Metric errors especially occur when doing manual estimations. Most software cost estimation tools support a LOC assumption for different programming languages, so that the estimation algorithms can be adjusted for each language.

**Impact of the Error**
When an estimate contains a metric error (e.g. for the productivity estimation by using the LOC from one language to estimate the LOC of another language) the deviation from the observed result can exceed 100 percent, if the languages vary widely.

### 3.2 Scaling Errors
**Error Description**
Scaling errors occur, if historical data from small projects is used as estimation database for large sized projects. Large projects differ from small ones mainly in two ways [Jones98]:

1. Large systems require more activities than small programs
2. The costs of large systems do not follow the same profile as small programs

**First issue:** The chart of accounts from Table 1.2 shows different activities that are performed for projects. Small projects often only perform at most 10 of 25 activities. Large civilian systems normally perform at least 20 of the 25 activities, and large military systems perform all 25 activities most of the time (see also Table 3.3, number of activities for different projects).
**Second Issue:** The effort for the activities are dependent on the size of the project (see Table 3.1) The size of the project is reflected by the function points in the table respectively by the KLOCs of code of the resulting system.
One reason, why scaling errors often occur is because the estimation of large projects is done only a few times by project managers in their whole career. Therefore the experience with large scale projects is missing. Scaling errors occur very often. For example in [Jones98] is mentioned a government project with about 15 million LOC that was estimated with data from projects of less than 500 function points (or about 15000 LOCs).

**Table 3.1:** Variantions in Software Effort Associated with Application Size [Jones98]

| Size, Function Poins | Size, KLOC | Coding % | Paperwork % | Defect Removal % | Management and Support % |
|---|---|---|---|---|---|
| 1 | 0.1 | 70 | 5 | 15 | 10 |
| 10 | 1.0 | 65 | 7 | 17 | 11 |
| 100 | 10.0 | 54 | 15 | 20 | 11 |
| 1000 | 100.0 | 30 | 26 | 30 | 14 |
| 10000 | 1000.0 | 18 | 31 | 35 | 16 |

**Impact of the Error**
Using data from small projects to estimate big projects can result in estimation errors of about one magnitude that means 1000 percent deviation.

### 3.3 Executive and Client Errors
**Error Description**
Senior executives or client executives influence project estimation that they have the corporate politics power to arbitrarily reject valid estimates of a software project. Original estimates which are accurate or at least conservative are therefore rejected by executives.

Often managers have the point of view, that the estimates do not fit in the financial or time scope of the enterprise. Therefore the project manager has to recast the estimate in order to get lower cost or a shorter time schedule.
Because of this, software costs often reflect the subjective opinions of the executives, not the accurate and more objective estimates of the project managers.
An analysis on cancelled software projects, which exceeded the time schedule or the project budget by more than 50 percent, approved that the cost estimates of more than 50% of these failed projects were influenced by executive managers or client executives and that the original estimates were accurate [Jones98]. Anyway, the project managers had still to take on responsibility.

**Impact of the Error**
In a project with client or executive errors, the direction of the errors is always the same: The projects take longer and cost more. Typical range of political errors result in a deviation of about 50 percent for the schedules and about 100 percent for costs.
This results can also be observed in practice, where often projects are twice as long as estimated and cost almost as much more as the forced estimate. We can also observe that the estimation of cancelled software projects are “manipulated” by executives or client executives.

### 3.4 Sizing Errors
**Error Description**
Predicting size is a very difficult task and often a very common source of error in software cost estimation. This can happen for external deliverables, such as

- Quantity of source code
- Number of screens
- Number of pages in user documentation
- Etc.

Estimation errors are also possible for internal deliverables like:

- Pages of specifications
- Pages of planning documents
- Test cases
- Etc.

The prediction of software cost for the above mentioned deliverables is supported by most tools which are today available. Size estimating errors are more common for manual estimates or when using older tools which do not support automated size estimation.

**Impact of the Error**
The deviation of the cost is approximately linear to the difference between the true size of the product and the anticipated size. There are empirical insights, that experienced project managers have a deviation in their size estimation of about plus or minus 15 percent, if the requirements are stable during the whole software process. The deviation of estimations of inexperienced project managers have big fluctuations in size errors and they approach deviations of about 100 percent and more.

### 3.5 Activity-Selection Errors
**Error Description**
Activity-selection errors occur when necessary work is omitted (example: Omitting the activity for writing a user documentation).
Most modern software cost estimation tools have a repository of more than 20 activities and corresponding tasks which are adjusted according to the project characteristics (e.g. if it is a

military or a civilian project). There is often the possibility to use templates of historical projects or customized templates for activity and task selection. The reasons for the occurring of this error is very often that manual estimation methods are used.
A possible pattern for a list of activities can be seen (see also [Jones98] and [Jones02]) in Table 3.2. This table contains all the different activities during a software project with all the effort which is spent for each activity (in relation to the overall project effort) depending on the project type. As we can see, there are large variations within the set of activities for each project type, therefore it is very dangerous to make rough estimations on the total project. Breaking down the estimation into activities and tasks, will end up in a much more reliable estimation of costs. It is also much more easy to validate the estimation because the inner structure of the performed work is apparent.
From the 1960s to the early the 1990s, phase level and project level estimates were the norm when estimating software costs. Since the beginning of the 1990s, the usage of activity level based estimates is more and more common.
Most tools for software cost estimation provide a standard list of activities and tasks and suggest an appropriate set of activities for a given project. More sophisticated tools will also differentiate different types of projects (civilian, military, large, small).

**Table 3.2:** Percentage of Staff Effort by Activity [Jones98]

| | Project Type in Percentage of Overall Project Effort | | | | | | |
|---|---|---|---|---|---|---|---|
| **Activity** | **End User** | **Web** | **MIS** | **Outsource** | **Commercial** | **System** | **Military** |
| Requirements | - | 3 | 7.5 | 9.0 | 4.0 | 4 | 7.0 |
| Prototyping | 10 | 10 | 2.0 | 2.5 | 1.0 | 2 | 2.0 |
| Architecture | - | - | 0.5 | 1.0 | 2.0 | 1.5 | 1.0 |
| Project plans | - | - | 1.0 | 1.5 | 1.0 | 2.0 | 1.0 |
| Initial design | - | - | 8.0 | 7.0 | 6.0 | 7.0 | 6.0 |
| Detail design | - | - | 7.0 | 8.0 | 5.0 | 6.0 | 7.0 |
| Design reviews | - | - | - | 0.5 | 1.5 | 2.5 | 1.0 |
| Coding | 35 | 25 | 20.0 | 16.0 | 23.0 | 20.0 | 16.0 |
| Reuse acquisition | 5 | 5 | - | 2.0 | 2.0 | 2.0 | 2.0 |
| Package purchase | - | - | 1.0 | 1.0 | - | 1.0 | 1.0 |
| Code inspection | - | - | - | - | 1.5 | 1.5 | 1.0 |
| Independent Verification and Validation | - | - | - | - | - | - | 1.0 |
| Configuration management | - | - | 3.0 | 3.0 | 1.0 | 1.0 | 1.5 |
| Formal integration | - | - | 2.0 | 2.0 | 1.5 | 2.0 | 1.5 |
| User documentation | 10 | 5 | 7.0 | 9.0 | 12.0 | 10.0 | 10.0 |
| Unit testing | 40 | 25 | 4.0 | 3.5 | 2.5 | 5.0 | 3.0 |
| Function testing | - | 17 | 6.0 | 5.0 | 6.0 | 5.0 | 5.0 |
| Integration testing | - | - | 5.0 | 5.0 | 4.0 | 5.0 | 5.0 |
| System testing | - | - | 7.0 | 5.0 | 7.0 | 5.0 | 6.0 |
| Field testing | - | - | - | - | 6.0 | 1.5 | 3.0 |
| Acceptance testing | - | - | 5.0 | 3.0 | - | 1.0 | 3.0 |
| Independent testing | - | - | - | - | - | - | 1.0 |
| Quality assurance | - | - | - | 1.0 | 2.0 | 2.0 | 1.0 |
| Installation and training | - | - | 2.0 | 3.0 | - | 1.0 | 1.0 |
| Project management | - | 10 | 12.0 | 12.0 | 11.0 | 12.0 | 13.0 |
| **Total** | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **Total number of Activities** | 5 | 8 | 18 | 21 | 20 | 23 | 25 |

### Impact of the Error
Activity-selection errors can vary widely and exceed an order of magnitude or 1000 percent. The worst case would be, if the estimate belongs only to the coding activity. Because in software development there is the tendency to see only the coding work, these errors are very common.

## 3.6 Assignment-Scope Errors
### Error Description
The assignment-scope means the quantity of work that can be handled by the staff. If the quantity of work is too high, there is an error in the assignment-scope.

Today all tools provide the possibility to make assignment-scope prediction for natural (e.g. lines of code, numbers of pages) and synthetic metrics (e.g. function points) or by user defined templates which are created based on historical project data. The error is more common for manual estimates. It results often if there are too few people available for the work. The error is more severe for maintenance estimates and for estimates of customer support. The error also depends on the capabilities of the staff. If there is staff available which is very experienced, it is possible that the same number of people can handle three times larger tasks than novices.
For making assignments of work, the estimator has to determine two features:

- Grouping of job profiles
- Estimating the work load that is handled on average by each job profile.

Empirical studies observed work scopes for the following occupation groups:

- Software development engineers and/or programmers
- Software maintenance engineers and/or programs
- System analysts
- Technical writers
- Quality-assurance specialists
- Configuration control specialists
- Integration specialists
- Testing specialists
- Customer-support specialists
- Project managers

In many large software companies like Microsoft, IBM, etc. there can be hundreds of such job profiles. There are also job profiles which contain more cross functionality in their job. Therefore the estimation for the assignment scope is very difficult to do. Examples for such job profiles are:

- Administrative specialists
- Database administration specialists
- Network specialists
- Multimedia specialists
- Human factors specialists
- System performance specialists
- Process specialists

An unsolved problem in the field of assignment scope research is that many companies do not use job profiles to categorize their software employees, or even have no job titles at all. Assignment-scope research is therefore very difficult.
Also one important fact is, that the assignment scope is becoming more and more important because of four recent trends [Jones98]:

- Downsizing and major layoffs of personnel in large companies.
- Business process reengineering.
- Shortages of software personnel brought on by huge changes and shifts throughout the whole software sector through normative, cultural and other economical or social changes, like the year-2000 problem or the software changes for the euro currency.
- Increasing frequency of the outsourcing arrangements, i.e. personnel which are transferred to the outsource company.

**Impact of the Error**
The range of uncertainty can reach up to 100 percent for assignment-scope errors.

### 3.7 Production-Rate Errors

#### Error Description

The production rate denotes the amount of work that can be completed by one person within a standard period of time (such as hour, work day, and so on).
Production-rate estimation is supported by the most available software cost estimation tools which allow the users to create templates from historical data. Errors therefore often result from too optimistic estimates done manually.
The estimation for production-rates can be expressed in natural metrics (like LOCs or pages) or in synthetic metrics (like function points). The latter is used the most, because synthetic metrics are additive along different activities. “Old” natural metrics like LOCs allow no aggregation and are not comparable along different activities, e.g. adding LOC and number of pages of user documentation or comparing them makes no sense. Productivity measures are better expressed in synthetic metrics.
Table 3.3 shows average production rates in function points for the defined 25 main activities.

**Table 3.3:** Production-rates in function points [Jones98]

| | Function Points per Month | | | | Work Hours per Function Point | | |
|---|---|---|---|---|---|---|---|
| **Activity** | **Minimum** | **Mode** | **Maximum** | | **Maximum** | **Mode** | **Minimum** |
| Requirements | 50.00 | 175.00 | 350.00 | | 2.64 | 0.75 | 0.38 |
| Prototyping | 25.00 | 150.00 | 250.00 | | 5.28 | 0.88 | 0.53 |
| Architecture | 100.00 | 300.00 | 500.00 | | 1.32 | 0.44 | 0.26 |
| Project plans | 200.00 | 500.00 | 1500.00 | | 0.66 | 0.26 | 0.09 |
| Initial design | 50.00 | 175.00 | 400.00 | | 2.64 | 0.75 | 0.33 |
| Detail design | 25.00 | 150.00 | 300.00 | | 5.28 | 0.88 | 0.44 |
| Design reviews | 75.00 | 225.00 | 400.00 | | 1.76 | 0.59 | 0.33 |
| Coding | 15.00 | 50.00 | 200.00 | | 8.80 | 2.64 | 0.66 |
| Reuse acquisition | 450.00 | 60.00 | 2000.00 | | 0.33 | 0.22 | 0.07 |
| Package purchase | 350.00 | 400.00 | 1500.00 | | 0.38 | 0.33 | 0.09 |
| Code inspection | 75.00 | 150.00 | 300.00 | | 1.76 | 0.88 | 0.44 |
| Independent Verification and Validation | 75.00 | 125.00 | 200.00 | | 1.76 | 1.06 | 0.66 |
| Configuration management | 1000.00 | 1750.00 | 3000.00 | | 0.13 | 0.08 | 0.04 |
| Formal integration | 150.00 | 250.00 | 500.00 | | 0.88 | 0.53 | 0.26 |
| User documentation | 20.00 | 70.00 | 100.00 | | 6.60 | 1.89 | 1.32 |
| Unit testing | 70.00 | 150.00 | 400.00 | | 1.89 | 0.88 | 0.33 |
| Function testing | 25.00 | 150.00 | 300.00 | | 5.28 | 0.88 | 0.44 |
| Integration testing | 75.00 | 175.00 | 400.00 | | 1.76 | 0.75 | 0.33 |
| System testing | 100.00 | 200.00 | 500.00 | | 1.32 | 0.66 | 0.26 |
| Field testing | 75.00 | 225.00 | 500.00 | | 1.76 | 0.59 | 0.26 |
| Acceptance testing | 75.00 | 350.00 | 600.00 | | 1.76 | 0.38 | 0.22 |
| Independent testing | 100.00 | 200.00 | 300.00 | | 1.32 | 0.66 | 0.44 |
| Quality assurance | 30.00 | 150.00 | 300.00 | | 4.40 | 0.88 | 0.44 |
| Installation and training | 150.00 | 350.00 | 600.00 | | 0.88 | 0.38 | 0.22 |
| Project management | 15.00 | 100.00 | 200.00 | | 8.80 | 1.32 | 0.66 |
| Cumulative results | 3375 | 6580 | 15600 | | 69.39 | 19.56 | 9.50 |
| Arithmetic mean | 135 | 263.2 | 624 | | 2.78 | 0.78 | 0.38 |

#### Impact of the Error

The impact of this error is difficult to generalize, because the production rates are highly specific to activities being performed and to the skill and experience of the personnel doing the work. The production-rate errors will normally be expressed as the difference between the true rate and the anticipated rate.

### 3.8 Creeping User Requirements

#### Error Description

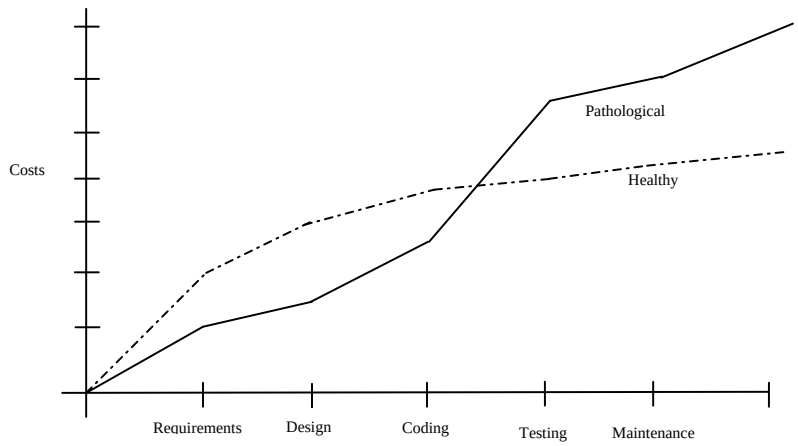
Creeping user requirements are the phenomenon of requirements which are not clear or not known when doing the initial requirements analysis. Creeping requirements begin to appear and evolve during the different phases of the software process. Empirical studies showed that creeping requirements are about 2 percent per month from the day of the initial agreement on requirements until the beginning of testing [Jones98]. Example: If there is a project which lasts about 24 month, the project schedule will grow roughly about 50 percent.

Creeping requirements have also to be taken into account when doing estimates for software projects. But omissions of creeping requirements are very common, especially when doing estimations manually. Most estimation tools support the estimation of creeping requirements. The function point metric provides a specially good possibility of predicting creeping requirements. One of the five parameters is almost certainly affected: Inputs, Output, Inquiries, Logical Files, Interfaces.
It's easy to calculate cost for additional functionality expressed in function points using a proportional or even a superproportional calculation scale.
Some of the today's tools support the prediction of a probable volume of creeping requirements.

**Impact of the Error**
The deviation for this error can be expressed as the difference of the volume between planned and unplanned functionality of the product, and as a thumb rule of estimation, it can be assumed that new requirements of about 2 percent of the initial requirements volume per month will be added or change existing requirements.

### 3.9 Critical Path Errors
**Error Description**
Software projects consist of a complex net of many activities. These activities contain many dependencies. Different tasks run often parallel and serial. In this network of activities, there is a path, which is critical, i.e. if one of the node activities of this path run late, the schedule of the whole project will be delayed.
The most frequent reason for critical path errors is related to debugging, testing and also other quality control measures. In 84 sample projects of IBM and ITT which ran late at least by six months [Jones98], there was too less time estimated for testing and debugging. The reason for this underestimation was mainly caused by executives and client executive errors which arbitrarily shortened the time schedule for the project. This approach very often causes problems.
The examination of these projects showed up all the same pathological behaviour. All projects were in the first phases on time and there was no evidence that the project was late, often the project seemed to be ahead of the project time table. When the testing phase started, lots of problems appeared suddenly, so that the project was delayed. Figure 3.1 shows the difference between a pathological and a healthy project. As conclusion one can say, that skimping in quality control measures results in late projects.

**Figure 3.1:** Project Characteristics of a Pathological and a Healthy Project [Jones98]



**Impact of the Error**
Empirical results show that this error causes a schedule delay of about 25 percent and cost deviation of about 35 percent. We also can assume that there are many costs hidden because of unpaid overtime which is very common for late projects.

### 3.10 Staffing Build-Up Errors
**Error Description**
This problem happens not very often, but if it does, it is quiet severe. If contracts on software projects are completed, often the personnel who is needed to do the project is not available in the own personnel pool or the personnel is involved in another project. Therefore the company has to recruit the needed additional staff after completion of the contract.
The staffing build-up errors occur, if the phase for the recruiting of the additional personnel takes longer than anticipated. This influences very often the time schedule of the project, so that the project can be delayed.
This problem is amplified if there is a shortage in the software personnel market (e.g. when the year-2000 problem had to be solved). As a result of this, software personnel salaries are growing faster and get onto a very high level, which also causes that beside a late project schedule the cost for the project will increase very fast to a much more higher level than estimated.

**Impact of the Error**
The range of uncertainty matches the difference between the planned and the available personnel. The error is not easy to predict, especially because the personnel market is coming into the play.

### 3.11 Technology Adjustment Errors
**Error Description**
Technology is always in change, i.e. methods for software engineering are continuously and newly developed. These methods and technologies are available in a huge range of tools. At the writing of [Jones98], the author states that more than 500 programming languages, more than 5000 software engineering tools, more than 150 specification and design methods and more than 50 different software methodologies were available on the market.
Technology adjustment errors occur when the influence of different used technologies is not correctly estimated. It is often very difficult to estimate the effect of applied technologies. The following examples describe possible sources of error when doing such estimates:

- Many software tools predict a productivity or quality increase without any empirical evidence. When project manager rely their estimates on such statements of software tool vendors, errors are possible.
- Often there are new methodologies and methods which are upcoming in software engineering, but the tools do not implement them, when software projects begin to use these methods or methodologies. Examples are the client/server or Java applications, which became popular before they were supported by estimation tools. A related problem is also that for these new technologies is missing the necessary historical data.
- Many new technologies have a very steep learning curve, which results very often in a much higher project effort than estimated (e.g. object-oriented software methods need much more time when they are used the first time in a project, compared to an old

method). Tool vendors and managers tend to forget that new technologies need some time to get productivity advantages.

The best is to stay current with advances in software technology. This should be done by the tool vendors, who should do a periodical update of the factors, which are included in the estimation tools.

**Impact of the Error**

The observed range of uncertainty for estimates can be up to 150 percent.

### 3.12 Special or Unique Situations

**Error Description**

A special or unique situation can have an effect on a specific project and cause that its schedule gets into a delay or the project exceeds the given cost budget. Such situations are often not easy or impossible to predict (and therefore do not fit into the estimation algorithms) or have only a very small probability of occurrence and can be handled as remaining risks. A few examples for special or unique situations are [Jones98]:

- Closure of an office or evacuation of staff due to weather conditions or fire or another natural disaster
- Voluntary termination of more than 50 percent of project team members
- Major layoffs or downsizing of project personnel
- Illness or incapacity of key team members
- Physical relocation of a project team from one city to another
- Injunctions or legal actions which freeze project specifications or source code
- Sale or merger of companies which affect specific projects
- Travel costs for trips among geographically dispersed projects
- Moving, living, and real-estate costs for hiring new employees
- Reassignment of key personnel to special problems (like the year-2000 repair work).

**Impact of the Error**

Usually there is no way of dealing with such problems ahead of time. Also cost and schedule deviations are not predictable and can range in a wide variety.

## 4. Summary and Conclusions

One precondition to get precise cost estimations is to get accurate cost data from former software projects, which can be used to determine the accuracy of an estimation or an estimation tool/method and to calibrate the estimation database. Cost data of former software projects is often biased and has therefore to be adjusted first.

During the estimation process, there are many possible sources of error. The following Table 4.1 should again summarize the errors and show their impact on the cost estimation in a short way:

**Table 4.1:** Summary of all Possible Sources of Error in Cost Estimation

| **Error** | **Impact on the Cost Estimation*** |
|---|---|
| Metrics Errors | Can exceed 100% |
| Scaling Errors | Up to 1000% |
| Executive and Client Errors | Up to 50% for schedules<br>Up to 100 % for costs |
| Sizing Errors | Between –15% and + 15% for experience estimators<br>Can exceed 100% for inexperienced estimators |
| Activity-selection Errors | Up to 1000 % |
| Assignment-scope Errors | Up to 100% |
| Productivity-rate Errors | ? |
| Creeping User Requirements | Increases about 2% of the size of the initial requirements every month |
| Critical path Errors | Up to 25% for schedules<br>Up to 35% for costs |

| | |
|---|---|
| Staffing build-up Errors | ? |
| Technology Adjustment Errors | Up to 150% |
| Special or Unique Situations | ? |

*****The impact on the software cost estimation is the deviation to the estimated project costs.

There are two possibilities to do software cost estimations. One possibility is to do it with a tool, the other is to do it manually. Manual estimations are very often too optimistic and therefore less accurate than estimations created with the help of tools. There are many factors which influence the costs of software projects. This plenty of data makes the manual estimation process very error-prone.
For example you can avoid activity-selection errors if you have some kind of checklist or guidance through the estimation process, which will help you to find the right activities for your project. Also many of the estimation errors listed in Table 4.1 occur mainly when doing manual software cost estimations. Tools can help to avoid making these errors or at least they can help to get accurate cost estimations that contain errors on a more moderate level.
But the usage of a software cost estimation tool is no guarantee for a good estimation, the estimator has still to have much experience and knowledge about former projects to get good results.

## 5. Appendix

### 5.1 References

[Jones98] Jones, C. (1998), Chapter 9 : “Estimating Software Costs”, New York: McGraw-Hill

[Jones02] Jones, C. (2002) : “Software Cost Estimation in 2002”, Utah: Software Technology Support Center, Hill Air Force Base, http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.pdf

### 5.2 English/German Terms

| English Term | German Term |
|---|---|
| to anticipate | erwarten |
| chart of account | Kontenplan |
| exempt | ausgemustert |
| funding | Finanzierung |
| to recast | umgestalten |
| to skimp | knausern, geizen |
| steep | steil |