Estimation Tools

Seminar on Software Cost Estimation
WS 02/03

Presented by
Christian Seybold
seybold@ifi.unizh.ch

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

Prof. M. Glinz
Arun Mukhija

Date: January 28, 2003

Table of Contents

Abstract

This talk is the last one in the series about Software Cost Estimation in the winter semester 2002/03. We heard a lot about different techniques how to measure costs and effort of software development. Up to now this was quite theoretical. This will change with this talk: it is about estimation tools. Apart from some statistics about estimation tools used in industry, the main focus lies on the demonstration of three selected tools. It concludes with some statements on the use of tools that are to be discussed. As the character of tools shall be practically demonstrated, this document differs in this point from the talk. In the talk, a demonstration of these tools will be given whereas in this document the tools are described from a theoretical point of view.

1 Introduction

Measurement using manual methods is difficult and expensive. For many years, the only effective measurement tools were proprietary ones built by various corporations for their own internal use. Starting about 20 years ago, a new sub industry began to emerge of companies that build measurement tools for software quality, complexity, defect tracking, cost tracking, schedule tracking, tools inventories, function point tracking, and many other measurement topics.

As of 2001, there are more than 50 commercial software cost estimation tools marketed in the United States, and at least 40 of them support function point metrics. The "best practice" for software estimation is to utilize one or more of these software estimation tools and have experienced managers, consultants, and technical personnel validate the estimate. For development schedules, output from most commercial software cost estimating tools can feed directly into project management tools although many commercial estimating tools also include schedule logic and can produce approximate schedules themselves [Jones98a].

The remainder is organized as follows. Chapter 2 categorizes software management tools and gives some interesting statistics about cost estimation tools in distinction to other categories. Chapter 3 describes three selected tools. Mainly, there functionality is listed and a short evaluation is given. The last chapter gives some statements on how to work with estimation tools. The provocative ones are to be discussed.

2 Statistics about software management tools

Software management tools include all kind of tools used in the context of software development. To be able to understand the capability and the use of these tools, we need to know which types of tools exists. So, in table 1 a categorization of tools is given with their main functionality. As project planning and cost estimation tools are the most important ones, these categories are described in more detail. Often, tools and categories cannot be mapped accurately. Many tools cover more than one category.

Tool category	**Functionality**
Project planning (project management)	• General purpose scheduling functions • Scheduling functions, Critical path analyses, precedence analysis, … • Resource management • Project tracking • Reporting functions, producing Gantt and PERT charts • Laying out the sequence of activities • Accumulating costs • Handling staff patterns • Lack of software projects knowledge base
Cost estimation	• One of the most important categories • Tools are based on knowledge of software projects • Full sizing capabilities, activity-based schedule, cost estimating capabilities, quality and reliability estimating capabilities, … • Function point and lines-of-code estimates including conversion in either direction • Side-by-side capability, show same project using a different development scenario • Schedule estimates usually to the phase levels, not to the scheduling of individual employees • Often includes quality estimates
Statistical analysis	• Provide statistical functionality

	• Most common: producing averages for productivity and quality by organization unit and by time period
Methodology management (process management)	• Usually keyed to one development methodologies, such as rapid application development • Provide guidance what deliverables are needed for scheduling, planning, … • Aggregate managerial functions under one umbrella
Year-2000 analysis	• Source code scanners, automated and semi-automated repair for the year-2000 problem
Quality estimation	• Prediction of defect potentials and defect removal efficiency • Estimation models are often company internal
Risk analysis	• Prediction of technical risks, sociological risks • Prediction of the probability that a project will run late, exceeds its budget, …
Portfolio analysis	• Management of the application portfolio of a company, e.g. to get the taxable value
Assessment support	• Support of standard forms of evaluating software methods and approaches, e.g. CMM
Project measurement	• Gather and manage project properties like size information of key deliverables, staffing, effort, cost data and defects.
Complexity analysis	• Measure complexity of source code, i.e. cyclomatic complexity, essential complexity, …
Value analysis	• Prediction of direct and indirect revenues, development, maintenance, marketing and support costs of software to be marketed • For in-house software, the value concepts are more difficult to enumerate
Budget support	• Oldest form of project management automation • Deals with continuous monthly expenditures for software development (they are not equal to departmental budget costs)
Variance-reporting	• Deal with monthly variances or differences between anticipated expenditures and actual expenditures
Project milestone tracking	• Tracking of resources, milestones or accomplishments • Record planned and actual events
Defect-tracking and –measurement tools	• Predict, track and analyze defects
Function point analysis	• Automate function point calculations • Assign function points to different business units • Derive function points from design or source code
Source code counting	• Parse or count source code

Tab 1: Software management tool categories and their functionality

In the following, several statistics are given comparing certain aspects of the software management tools categories used in industry. They are about sizes of the used tools, costs and benefit. All tables except the first one are taken from [Jones98a]. The first one is taken from [Jones98b], as the numbers are more up to date there. However, as they date from 1999, the "Year 2000 analysis" category is still included, though it does not play a role any more in our days. The categories can vary slightly from table to table as they are taken out of different contexts. After that, some statements are given on the training aspects of people going to work with tools.

2.1 How many tools do you need?

Table 2 [Jones98b] shows for lagging, average and leading companies, how meaningful each software management tool category is for them. The importance factor is expressed in function points.

You can see that even lagging companies are using a considerable amount of project planning tools whereas cost estimating tools, that have the same size for leading companies, don't play any role for lagging or average companies.

Category	Lagging	Average	Leading
Project planning	Fp 1,000	Fp 1,250	Fp 3,000
Project cost estimating			3,000
Statistical analysis			3,000
Methodology management		750	3,000
Year 2000 analysis			2,000
Quality estimation			2,000
Assessment support		500	2,000
Project measurement			1,750
Portfolio analysis			1,500
Risk analysis			1,500
Resource tracking	300	750	1,500
Value analysis		350	1,250
Cost variance reporting		500	1,000
Personnel support	500	500	750
Milestone tracking		250	750
Budget support		250	750
Function point analysis		250	750
Backfiring: LOC to FP			750
Function point subtotal	**1,800**	**5,350**	**30,250**

Tab 2: Sizes of project management tools (in FP) used in lagging, average and leading companies (1999).

2.2 How much do these tools cost?

In table 3 you can see the average costs for one seat (one user) of these tools for three different price levels. So, cost estimating tools are far the most expensive ones as they are based on complex models. Quality estimating tools are second most expensive. Project planning tools are somewhere in the middle field, as there is nothing complicated about them. If there are costs of 0$ shown, this means that there are free tools available. Looking at the costs only is not meaningful. You should also take the benefit into account. This is done in the following table 4.

Tool-Category	Low-cost	Median-cost	High-cost
Cost estimating	$250	$2.500	$15.000
Quality estimating	150	500	5.000
Methodology management	0	1.250	3.500
Complexity analysis	75	500	3.000
Statistical analysis	0	750	2.500
Assessment support	0	750	2.000
Project planning	100	500	1.500
Risk analysis	0	500	1.500
Value analysis	0	500	1.500
Portfolio analysis	0	750	1.500
Year 2000 analysis	350	750	1,500
Project measurement	75	500	1.250
Budget support	0	500	1.000
Cost variance reporting	0	500	1.000
Project tracking	100	300	1,000
Defect tracking	200	500	1.000
Function point analysis	50	500	750

Source code counting	0	250	500
Total	**$1.350**	**$12.300**	**$45.000**
Average	**$75**	**$683**	**$2.500**

Tab 3: Approximate cost per seat for software management tools

2.3 How much benefit do they provide?

The four-year return on investment (ROI) for each dollar tool investment is shown in table 4. It is given for each of the four years as well as the total sum over the four years. Remember in the previous table, cost and quality estimating tools were the most expensive ones. Here, you can see, that this investment is nevertheless worth, because you can still get the highest benefit of them.
All tools have more or less strongly increasing ROI over the years with one exception: the year-2000 analysis tools. Dating from 1999, there was only one year left to the millennium change, so that the benefit of these investments shortly was to return. This explains this amazing high number. But, this was a single event and we don't expect a similar event of this size. Already three years later, there is no RIO at all any more.

Tool-Category	Year 1	Year 2	Year 3	Year 4	Total
Quality estimating	$3.00	$4.50	$12.00	$18.00	$37.50
Cost estimating	2.50	5.00	12.00	17.50	37.00
Methodology management	2.00	3.50	10.00	16.00	31.50
Project measurement	1.50	3.50	8.50	13.00	26.50
Project planning	1.50	4.00	8.00	12.50	26.00
Defect tracking	2.00	3.00	7.00	13.00	25.00
Year 2000 analysis	*15.00*	*5.00*	*2.00*	*0.00*	*22.00*
Assessment support	1.50	3.00	6.00	12.00	20.50
Project tracking	1.75	3.50	6.00	10.00	20.50
Value analysis	1.50	3.00	5.00	8.00	17.50
Function point analysis	1.75	3.00	4.50	8.00	17.25
Portfolio analysis	1.75	2.50	5.00	7.50	16.75
Statistical analysis	1.75	2.75	4.50	6.00	15.00
Risk analysis	1.50	2.50	3.50	5.50	13.00
Complexity analysis	1.30	2.00	3.00	4.50	10.80
Cost variance reporting	1.00	1.50	2.50	3.50	8.50
Budget support	1.00	1.50	2.00	3.00	7.50
Source code counting	1.00	1.00	1.00	1.00	4.00
Total	**$43.30**	**$54.75**	**$102.50**	**$156.00**	**$356.55**
Average	**$2.41**	**$3.04**	**$5.69**	**$8.67**	**$19.81**

Tab 4: Approximate four-year return on investment in software management tools

2.4 Training aspects

Most complex tools require substantial training and sometimes consulting assistance to be used effectively. A study by Hewlett Packard of computer-aided software engineering (CASE) tools found that unless $1.00 was spent on training users for every $1.00 spent on CASE tools themselves, the tools did not improve performance significantly.

A general rule of thumb for software training is that users of application packages need about one day of instruction for every 3000 function points in the package. Thus for a typical user of SAP features such as an accountant or controller who will utilize about 30,000 function points (cp table 5), perhaps 10 days of training may be needed to get fully up to speed.

Feature	**Function Points**
Core SAP Integration	10,000
Financial accounting	50,000
Controller features	35,000
Fixed assets	25,000
Sales distribution	25,000
Materials management	30,000
Production planning	30,000
Quality management	20,000
Plant management	25,000
Human resources	15,000
Workflow	10,000
Project management	5,000
Cost modeling	2,500
Business engineering	5,000
TOTAL	287,500

Tab. 5: Subcomponents of the SAP software and its function point sizes

3 Cost Estimation Tools

As we have seen, cost estimating tools are the most important and the most expensive tools, but they are also the tools to expect the highest benefit. For the estimation process, they employ one or more of several known methods: parametric modeling, knowledge-based modeling, rule induction, fuzzy logic, dynamic modeling, neural networks, or case-based reasoning.
The Survey of the NASA about estimation tools [Seeds] gives a good overview about the unmanageable amount of existing tools and their qualities. Here, „only“ three tools are discussed in more detail. They are selected for following reasons:

- Availability, e.g. evaluation versions,
- a convincing first impression and
- a combination of several methods.

There are also tools available concentrating just on one method, e.g. function point analysis. But as these methods are already well known and a tool calculating the formula is not very exciting, they are not considered here. Finally, the following tools have been selected for further evaluation in the following sections:

- Construx Estimate Professional is mainly a software cost estimation tool based on a combination of estimation models.
- Tassc Estimator covers the software development process from architectural estimates up to staff scheduling.
- The UML modeling tool EnterpriseArchitect includes an interesting early estimate based on Use Case Metrics.

The locations where these tools can be obtained are given in the reference section at the end of this document. In the following sections, the tools are introduced in more detail. But before that, a brief excursion is given on accuracy and precision and what this means for tools.

3.1 Accuracy and precision

Accuracy and precision are related but not identical concepts, and the difference between the two is important to software estimation. Accuracy refers to how close to the mark a measurement is. 3 is more accurate representation of pi than 4 is. (To seven decimal places, pi equals 3.1415927.) Precision refers to how many significant digits a measurement has. 3.14 is a more precise representation of pi than 3 is.
A measurement can be precise without being accurate, and it can be accurate without being precise. 3 is an accurate representation of pi, but it is not precise. 3.3232 is a precise representation of pi, but it is not accurate. Airline schedules are usually precise to the minute, but they are not very accurate. Measuring people's heights in whole feet might be accurate, but it would not be precise.
In software estimation, false precision is the enemy of accuracy. An effort estimate of 40 to 70 man-months might be both the most accurate and the most precise estimate you can make. If you simplify it to 55 man-months, that's like representing pi as 3.3232 instead of 3. It looks more precise, but it's really less accurate.
Shortest-possible software schedules are achieved by creating the most accurate estimates possible, not the most precise. If you want to achieve maximum development speed, avoid false precision [Construx].

3.2 Construx Estimate Professional

Construx Software Builders[1] provide Construx Estimate [Construx], a free estimation that includes the functionality of both COCOMO II and SLIM. Construx Estimate uses Monte Carlo simulations to model complex interactions in the face of uncertain estimating assumptions, making the company one of the few who offer a stochastic method. On their website, they describe their tool themselves: "With over 30,000 users, Construx Estimate is the most popular estimation software in the world. Construx Estimate is a parametric estimation tool that provides better predictability of your projects". Its features are listed in table 6.

Feature	Description
Sophisticated Modeling	Estimate utilizes Monte Carlo simulation with 2 powerful estimation models (SLIM and COCOMO) to model the cone of uncertainty.
Historical Data Calibration	Achieve pinpoint estimation accuracy by calibrating Estimate with projects from your own organization and project teams.
Industry Data	If you don't have historical data for your project, use Estimate to develop rough estimates based on industry performance data for different types of applications.
GUI project estimation	You can create estimates very early in a project by counting the number of dialog boxes, reports, graphical outputs, and so on.
Function Point and Lines of Code	Estimate supports direct entry of both Function Points and Lines of Code to create detailed effort and schedule estimates.

[1] Founded by Steve McConnell. He served as Editor in Chief of IEEE Software from 1999 to 2002, was editor of IEEE Software's "Best Practices" column from January 1996 to August 1998, and acted as a senior reviewer for IEEE Computer magazine. He is a member of IEEE Computer Society and ACM.

Multi-module estimation	Create size estimates for multiple modules within a project -- modules written in different languages, that are built by separate teams, or that are broken into separate subsystems or components.
Reports	Estimate contains 19 reports that describe your estimate in detail. Explain your estimate to your management, clients, or teammates.
Constraints and Priorities	Estimate allows you to enter constraints on cost, schedule, peak staff, and maximum effort allowed. You can also set relative priorities for cost, schedule, peak staff, and maximum effort.

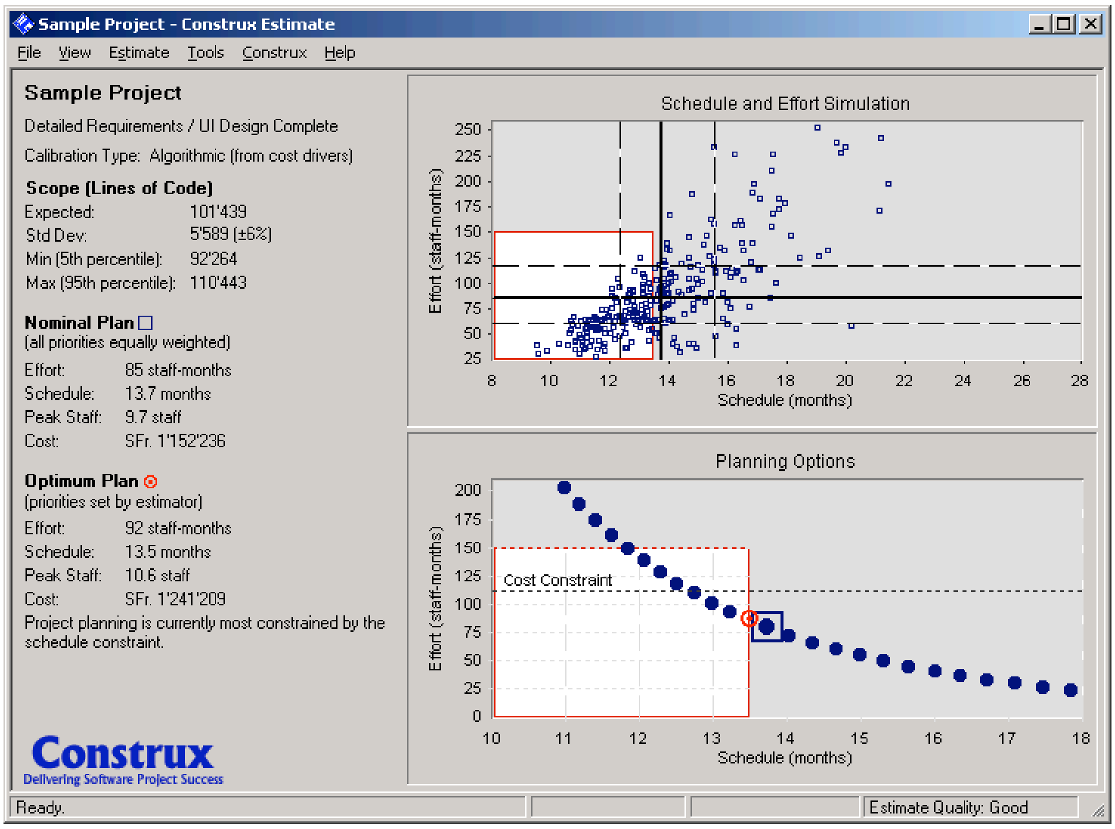
Tab. 6: Functionality of the Construx Estimate Professional Tool

Construx Estimate makes use of three mature estimation approaches:

1. SLIM was developed by Lawrence H. Putnam in the early 1970s and first offered as a commercial product in 1978. The SLIM methodology is based on the insight that efficiently run software projects follow well-defined patterns that can be modeled with a set of exponential equations. These equations form the backbone of Construx Estimate's approach to creating cost, schedule, peak staffing, and defect estimates.

2. Cocomo 2.0 allows estimates to be created for virtually any kind of project by specifying a set of cost drivers. Construx Estimate uses the Cocomo 2.0 model as a supplement to the SLIM model when estimates are calibrated using cost drivers. A productivity baseline is established using the project type settings; the productivity factor is then adjusted using the computed Cocomo 2.0 productivity. Construx Estimate uses Cocomo 2 data and algorithms from Cocomo II Model Definition Manual, version 1.4.

3. Construx Estimate uses Monte Carlo simulations to model complex interactions in the face of uncertain estimating assumptions. Construx Estimate simulates hundreds or thousands of possible outcomes of the project being estimated based on size, productivity, current project phase, and other parameters entered by the estimator. It then estimates the likelihood of various project outcomes and assigns risk levels to different planning options. In complex situations that involve a lot of uncertainty, the methodology allows Construx Estimate to create meaningful estimates that would otherwise be impossible to model.


Sample Project - Construx Estimate
File View Estimate Tools Construx Help
Schedule and Effort Simulation
Effort (staff-months)
Schedule (months)
Planning Options
Cost Constraint
Construx
Delivering Software Project Success

Sample Project

Detailed Requirements / UI Design Complete

Calibration Type: Algorithmic (from cost drivers)

Scope (Lines of Code)

Expected:	101'439
Std Dev:	5'589 (±6%)
Min (5th percentile):	92'264
Max (95th percentile):	110'443

Nominal Plan □

(all priorities equally weighted)

Effort:	85 staff-months
Schedule:	13.7 months
Peak Staff:	9.7 staff
Cost:	SFr. 1'152'236

Optimum Plan ⊙

(priorities set by estimator)

Effort:	92 staff-months
Schedule:	13.5 months
Peak Staff:	10.6 staff
Cost:	SFr. 1'241'209

Project planning is currently most constrained by the schedule constraint.

Ready.

Estimate Quality: Good

Fig. 1: Screenshot of the Construx Estimate Professional Tool, data of a sample project

In fig. 1, you can see the main screen of the tool. It focuses mainly on estimation of cost, effort, schedule and peak staff providing both, a nominal and an optimal plan. It makes a good impression, is easy to use, well documented and it includes a lot of background information on software estimation as well as interesting case studies.

3.3 Tassc Estimator

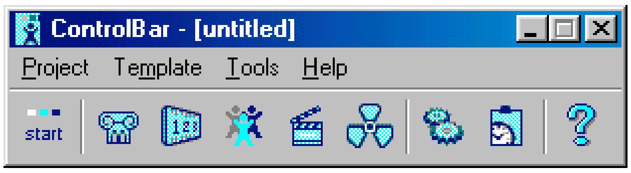


Fig. 2: Main menu of the Tassc Estimator Manager Edition.

The Tassc Estimator 2001 [Tassc] is available in three editions: lite, engineer and manager edition. They differ just in the amount of functionality. The lite edition is restricted to cost estimation based on architectural aspects whereas the manager edition also includes risk management and project scheduling. In this context, the manager edition is used (see fig. 2).

The estimation follows a step-by-step process. In the main menu, this is reflected by the horizontal alignment of the icons; in the model (see fig. 2) this is symbolized by the gray stations (see fig. 3). For each station, you can see which factors influence the estimation. Nevertheless, there is no need to work on them sequentially. At any point, the current estimation results can be viewed. An example is given in fig. 4. This fact supports both, a fast first guess as well as continual refinement of the input factors.

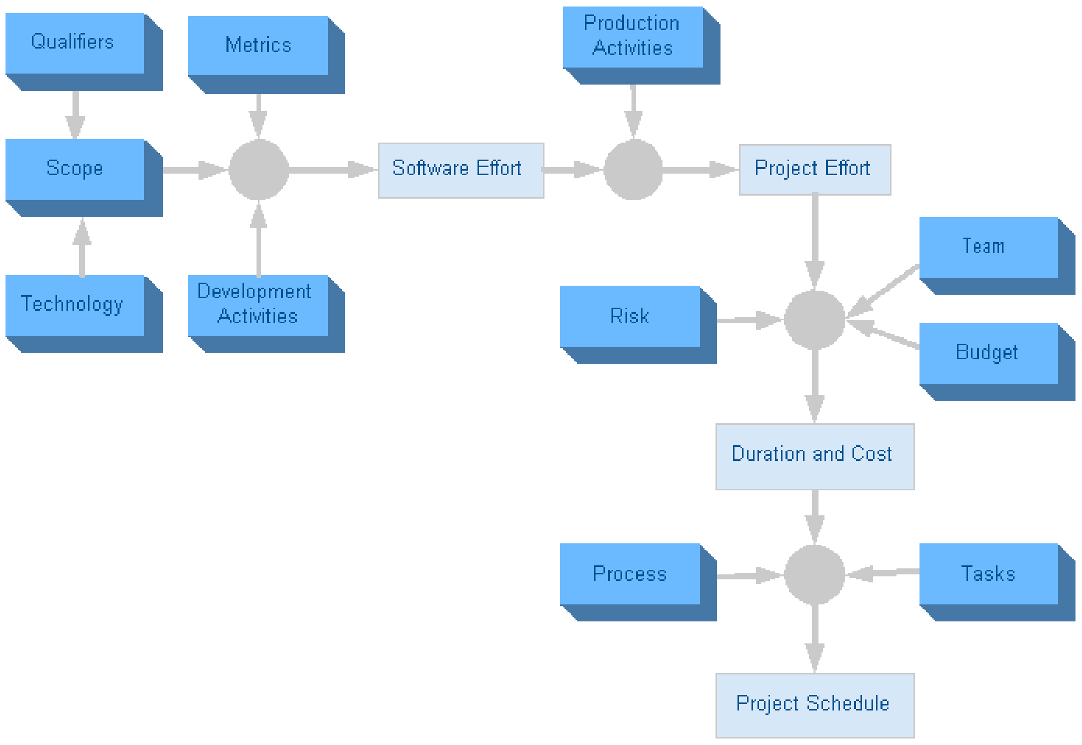
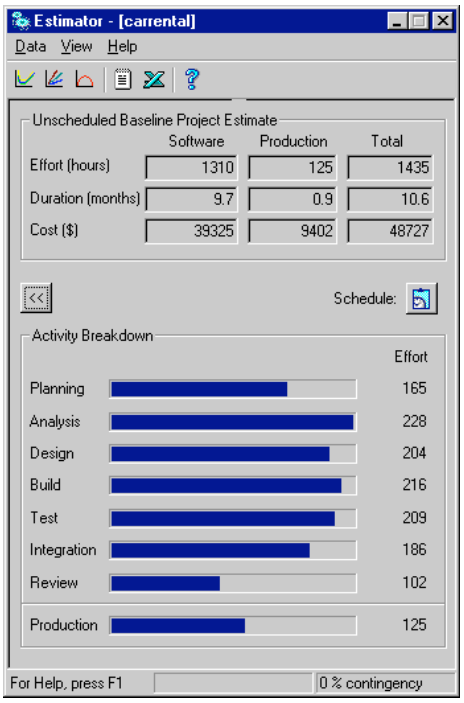


Fig. 3: Estimation process of the Tassc Estimator.

Beyond the pure cost estimation, this tool can deal with the whole development process, as team and scheduling functions are included. This makes it easy to estimate also remaining tasks and to compare the estimated numbers with the actual ones.
A nice tutorial on how to work with this tool is included whereas the documentation is not very detailed, especially concerning the estimation model.



Estimator - [carrental]

Data View Help

Unscheduled Baseline Project Estimate

	Software	Production	Total
Effort (hours)	1310	125	1435
Duration (months)	9.7	0.9	10.6
Cost ($)	39325	9402	48727

Schedule:

Activity Breakdown

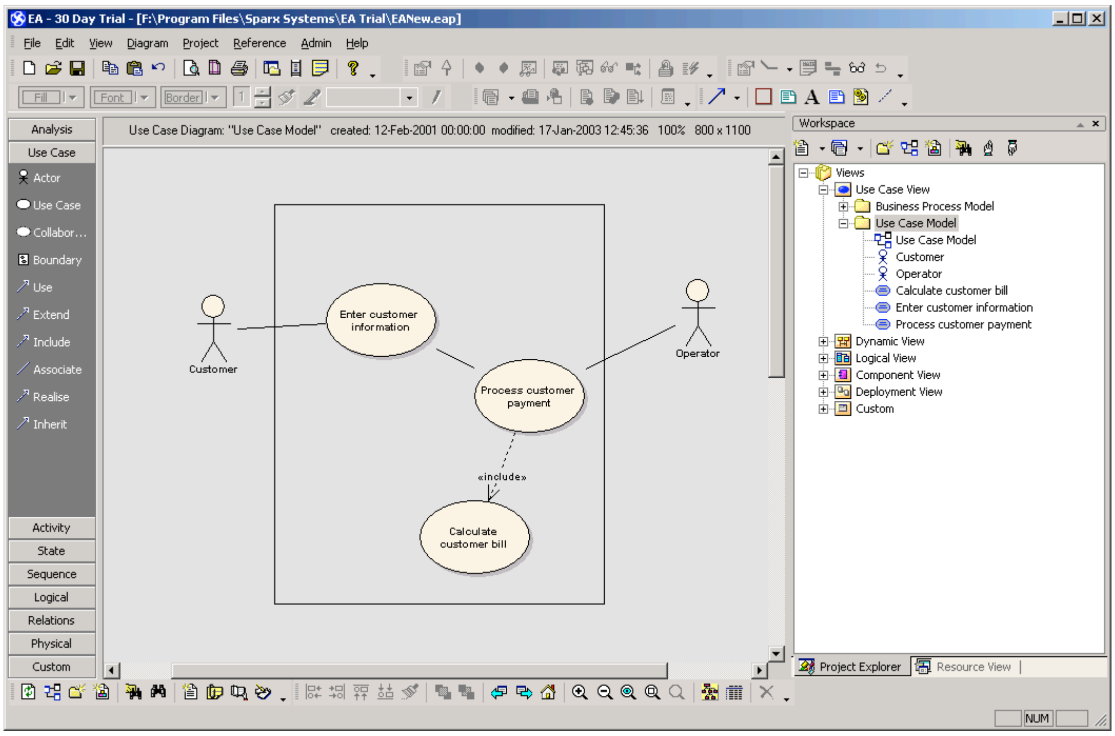
	Effort
Planning	165
Analysis	228
Design	204
Build	216
Test	209
Integration	186
Review	102
Production	125

For Help, press F1 | 0 % contingency

Fig. 4: Results screen of the Tassc Estimator.

3.4 Sparx Systems Enterprise Architect

EnterpriseArchitect [Sparx] is a fully functional UML modeling tool. Why it is chosen in this context is the originality of the provided use case metrics. Use case metrics can give a very first estimate based on use case diagrams. They are not supposed to be very accurate. But as they are produced very easily, they can help to calculate the effort in a very early phase of the project, e.g. to determine the feasibility of a project.

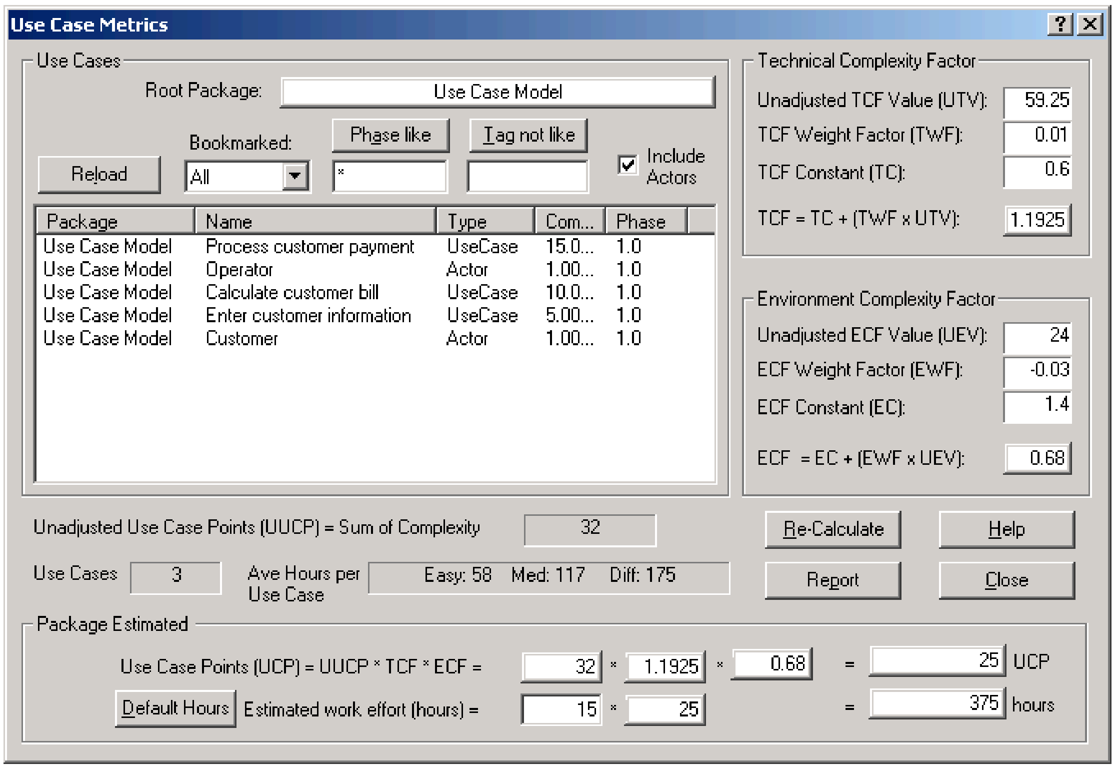


EA - 30 Day Trial - [F:\Program Files\Sparx Systems\EA Trial\EANew.eap]

Use Case Diagram: "Use Case Model" created: 12-Feb-2001 00:00:00 modified: 17-Jan-2003 12:45:36 100% 800 x 1100

Fig. 5: Enterprise Architect Use Case diagram.

In fig. 5 you can see the main screen of this tool. An example use case diagram is selected based on which the Use Case Metrics are calculated. The calculation form is given in fig. 6. First, unadjusted use case points (UUCP) are calculated based on the number and complexity of the drawn use cases. Multiplying this number with the technical and environment complexity factor yields the Use Case Points (UCP). Both complexity factors are themselves based on several weighted influence factors that are shown in the dialog windows in fig. 7. Finally, multiplying the UCP with a default hour rate per use cases gives the estimate effort. Without calibration of this default hour rate, the results won't be very meaningful.



Use Cases

Root Package: Use Case Model

Bookmarked: All

Phase like: *

Include Actors: ✓

Package	Name	Type	Com...	Phase
Use Case Model	Process customer payment	UseCase	15.0...	1.0
Use Case Model	Operator	Actor	1.00...	1.0
Use Case Model	Calculate customer bill	UseCase	10.0...	1.0
Use Case Model	Enter customer information	UseCase	5.00...	1.0
Use Case Model	Customer	Actor	1.00...	1.0

Technical Complexity Factor

Unadjusted TCF Value (UTV):	59.25
TCF Weight Factor (TWF):	0.01
TCF Constant (TC):	0.6
TCF = TC + (TWF x UTV):	1.1925

Environment Complexity Factor

Unadjusted ECF Value (UEV):	24
ECF Weight Factor (EWF):	-0.03
ECF Constant (EC):	1.4
ECF = EC + (EWF x UEV):	0.68

Unadjusted Use Case Points (UUCP) = Sum of Complexity: 32

Use Cases: 3

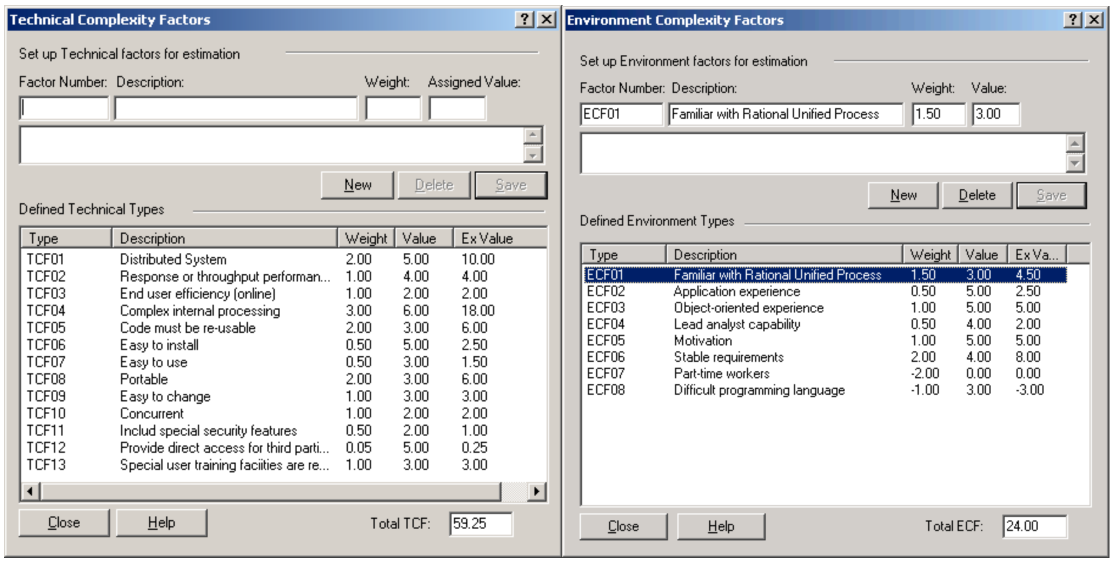
Ave Hours per Use Case: Easy: 58 Med: 117 Diff: 175

Package Estimated

Use Case Points (UCP) = UUCP * TCF * ECF = 32 * 1.1925 * 0.68 = 25 UCP

Estimated work effort (hours) = 15 * 25 = 375 hours

Fig. 6: Enterprise Architect Use Case Metrics Mask.



Set up Technical factors for estimation

Factor Number: Description: Weight: Assigned Value:

Defined Technical Types

Type	Description	Weight	Value	Ex Value
TCF01	Distributed System	2.00	5.00	10.00
TCF02	Response or throughput performan...	1.00	4.00	4.00
TCF03	End user efficiency (online)	1.00	2.00	2.00
TCF04	Complex internal processing	3.00	6.00	18.00
TCF05	Code must be re-usable	2.00	3.00	6.00
TCF06	Easy to install	0.50	5.00	2.50
TCF07	Easy to use	0.50	3.00	1.50
TCF08	Portable	2.00	3.00	6.00
TCF09	Easy to change	1.00	3.00	3.00
TCF10	Concurrent	1.00	2.00	2.00
TCF11	Includ special security features	0.50	2.00	1.00
TCF12	Provide direct access for third parti...	0.05	5.00	0.25
TCF13	Special user training faciities are re...	1.00	3.00	3.00

Total TCF: 59.25

Set up Environment factors for estimation

Factor Number: ECF01 Description: Familiar with Rational Unified Process Weight: 1.50 Value: 3.00

Defined Environment Types

Type	Description	Weight	Value	Ex Va...
ECF01	Familiar with Rational Unified Process	1.50	3.00	4.50
ECF02	Application experience	0.50	5.00	2.50
ECF03	Object-oriented experience	1.00	5.00	5.00
ECF04	Lead analyst capability	0.50	4.00	2.00
ECF05	Motivation	1.00	5.00	5.00
ECF06	Stable requirements	2.00	4.00	8.00
ECF07	Part-time workers	-2.00	0.00	0.00
ECF08	Difficult programming language	-1.00	3.00	-3.00

Total ECF: 24.00

Fig. 7: Technical and Environment Complexity Factors of the Use Case Metrics.

4 Conclusion

Software estimation is now sophisticated enough so that a formal estimate using one or more of the 50 commercial software estimation tools in conjunction with software project management tools can minimize or eliminate unpleasant surprises later due to schedule slippages or cost overruns. Indeed, old-fashioned purely manual cost and schedule estimates for major software contracts should probably be considered an example of professional malpractice. Manual estimates are certainly inadequate for software contracts or outsource agreements whose value is larger than about $500,000 [Jones00].

The NASA describes their lessons learned like the following [Seeds]:

- There is a minimum development time below which a system cannot be completed successfully.
- There is a useful trade-off between time and effort.
- There is a functional coupling between size, schedule, effort, and reliability – change one, the others all change.
- There is great payoff from improving the productivity of the development process.
- There is no silver bullet.
- New development methods change the way you size a project.

And finally, here are some hints from the author on the usage of software estimation tools:

- Using a tool is half the way for good estimates (just because you have recognized the problem, you start working systematically, gathered data is always available, and it is refined continually).
- There is no tool owning all functionality (don't wait for the silver bullet).
- Start using tools early (to gather experience).
- Use more than one tool at the same time (to equalize lacks and differences).
- Avoid the change of tools (to be not confused).
- There is no need to know all these formulas. But know your tool well!
- Don't rely on tool results without thinking about it.
- Remember the difference between accuracy and precision.
- Tools usually are more conservative and accurate than manual estimates.

References

[Construx]	http://www.construx.com/resources/estimate/
[Jones98a]	Jones, T.C. (1998). Estimating Software Costs. New York: McGraw-Hill.
[Jones98b]	Jones, T.C. (1998). Project management Tools and Software Failures and Successes. Crosstalk, The Journal of Defense Software Engineering. July 1998, p.13.
[Jones00]	Jones T.C. (2000). Conflict & Litigation between software clients & Developers. http://www.spr.com.
[Seeds]	NASA. SEEDS: Strategic Evolution of ESE Data Systems. Survey of Cost Estimation Tools. Deliverable 34.05.01, November 30, 2001.
[Sparx]	http://www.sparxsystems.com.au
[Tassc]	http://www.tassc-solutions.com