Software Engineering Economics

ACRONYMS TABLE

EVM	Earned Value Management
IRR	Internal Rate of Return
MARR	Minimum Acceptable Rate of Return
PLC	Product Life Cycle
ROI	Return on Investment
ROCE	Return on Capital Employed

Introduction

Software Engineering Economics is about making decisions related to software engineering in a business context. The success of a software product, service, and solution depends on good business management. Yet, in many companies and organizations, software business relationships to software development and engineering remain vague. This entry provides an overview on software engineering economics.

Software engineering economics means aligning software technical decisions with the business goals of the organization. In all types of organizations—be it “for-profit,” “not-for-profit,” or governmental—this translates into sustainably staying in their business. In “for-profit” organizations this additionally relates to achieving a tangible return on the invested capital—both assets and capital employed. This knowledge area has been formulated in a way to address all types of organizations independent of focus, product and service portfolio, or capital ownership and taxation restrictions.

Decisions like “Should we use a specific component?” may look easy from a mere technical perspective but can have serious implications on the business viability of the project and the resulting product.

This knowledge area first underlines the foundations, key terminology and basic concepts, and practices; thus, it indicates how decision-making in software engineering gets a business perspective. It starts with providing the foundations on software engineering economics, then evolves to a life-cycle perspective, highlights risk and uncertainty management, and shows how economic analysis methods are used. Some practical considerations close the knowledge area (see Figure 1 for the breakdown).

Breakdown of Topics for Software Engineering Economics

1. Software Engineering Economics Fundamentals

1.1. Finance

Finance is a branch of economics that deals with resource allocation and management, acquisition, and investment. The field of finance deals with the concepts of time, money, risk, and how they are interrelated. It also deals with how money is spent and budgeted. Corporate finance is the task of providing the funds for an organization’s activities. It generally involves balancing risk and profitability while attempting to maximize an organization's wealth and the value of its stock. This holds primarily for “for-profit” organizations, but also for “not-for-profit” organizations. The latter need finance to ensure sustainability while not targeting tangible profit. To do this, an organization must:

- Identify and implement relevant business objectives and constraints, such as organizational or individual goals, time horizon, risk mitigation, and tax considerations;
- Identify and implement the appropriate business strategy, such as which portfolio and investment decisions to take, how to manage cash flow, and where to get the funding;
- Measure the financial performance, such as cash flow and ROI, and take corrective actions in case of deviation from objectives.

1.2. Accounting

Accounting is related to finance. It allows people whose money is being used to run the organization know the results of their investment: did they get the profit they were expecting? In “for-profit” organizations, this relates to the tangible ROI, while in “not-for-profit” and governmental organizations as well as “for-profit” organizations, it translates into sustainably staying in their business. The primary role of accounting is to measure the organization’s actual financial performance. Its purpose thus is to communicate financial information about a business entity to users such as shareholders. Communication is generally in the form of financial statements that show in money terms the economic resources to be controlled. It is important to select the right information that is relevant to the user and is reliable. Partially, the information and its timing is governed by risk management and governance policies. Accounting systems are also a rich source of historical data for estimating.

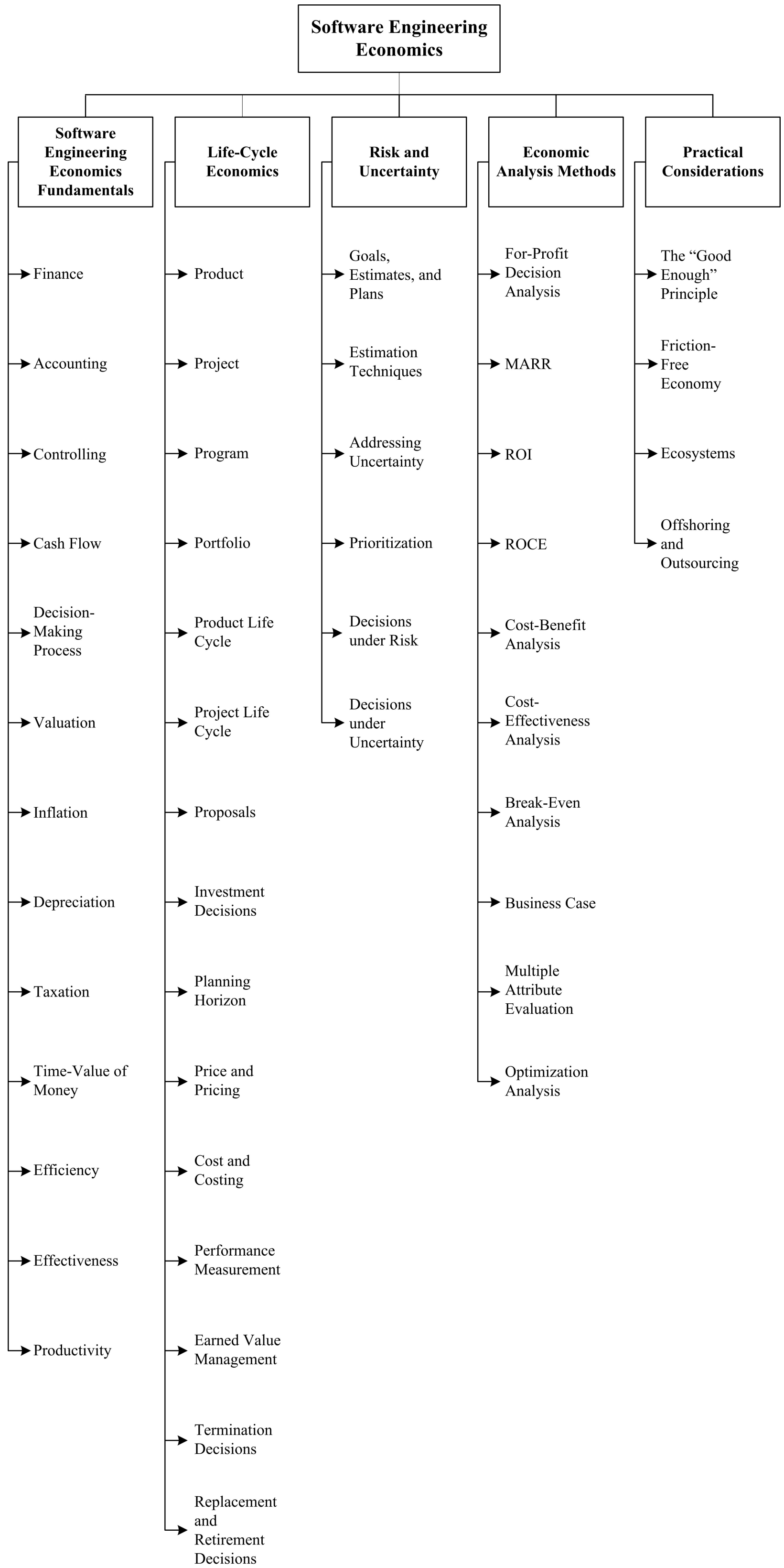

Figure 1: Breakdown of topics for the Software Engineering Economics Knowledge Area

1.3. Controlling

Controlling is part of finance and accounting. Controlling is the measurement and correction of performance in order to make sure that an organization's objectives and the plans devised to attain them are accomplished. Cost controlling is a specialized branch of controlling that is used to detect variances of the actual costs from the planned costs.

1.4. Cash Flow

Cash flow is the movement of money into or out of a business, project, or financial product over a given period. A *cash flow instance* is a specific amount of money flowing into or out of the organization at a specific time as a direct result of some activity.

The concepts of cash flow instances and cash flow streams are used to describe the business perspective of a proposal. To make a meaningful business decision about any specific proposal, that proposal will need to be evaluated from a business perspective. In a proposal to develop and launch product X, the payment for new software licenses could be an example of an outgoing cash flow instance. Money would need to be spent to carry out that proposal. The sales income from product X in the 11th month after market launch could be an example of an incoming cash flow instance. Money would be coming in because of carrying out the proposal.

The term *cash flow stream* refers to the set of cash flow instances, over time, that would be caused by carrying out some given proposal. The cash flow stream is, in effect, the complete financial picture of that proposal. How much money goes out? When does it go out? How much money comes in? When does it come in? Simply, if the cash flow stream for Proposal A is more desirable than the cash flow stream for Proposal B, then—all other things being equal—the organization would be better off carrying out Proposal A than Proposal B. Thus, the cash flow stream is an important input for investment decision-making.

A *cash flow diagram* is a picture of a cash flow stream. It gives the reader a very quick overview of the financial picture of that subject. Figure 2 shows an example cash flow diagram for a proposal.

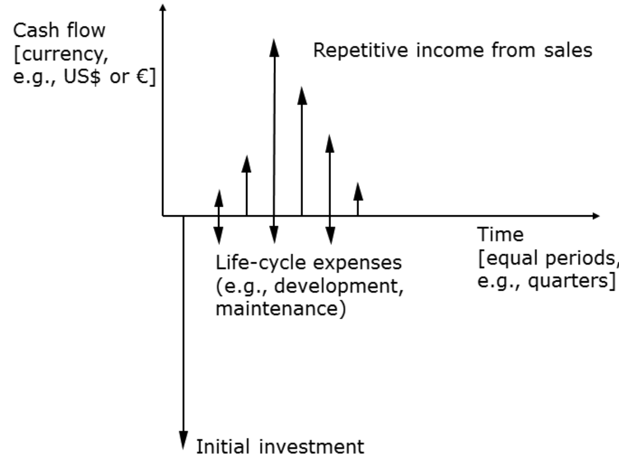

Figure 2: A cash-flow diagram

A cash flow diagram shows the cash flow stream in two dimensions, time runs from left to right and amounts of money run up and down. Each cash flow instance is drawn on the diagram at a left-to-right position relative to the timing of that cash flow after the start of the proposal. The horizontal axis is divided into units of time that represent either years, months, weeks, etc., as appropriate for the proposal being studied.

1.5. Decision-Making Process

If we assume that candidate solutions solve a given technical problem equally well, why should the organization care which one is chosen? The answer is that there is usually a large difference in the costs and incomes from the different solutions. A commercial, off-the-shelf, object-request broker product might cost a few thousand dollars, but the effort to develop a homegrown service that gives the same functionality could easily cost several hundred times that. If the candidate solutions all adequately solve the problem from a technical perspective, then the one that maximizes the return on the organization's investment is the one that should be chosen. To do this, the software engineer should follow a systematic process for making decisions. That systematic process is shown in Figure 3. It starts with a business challenge at hand and describes the steps to identify alternative solutions, define selection criteria, evaluate the solutions, implement one selected solution, and monitor the performance of that solution.

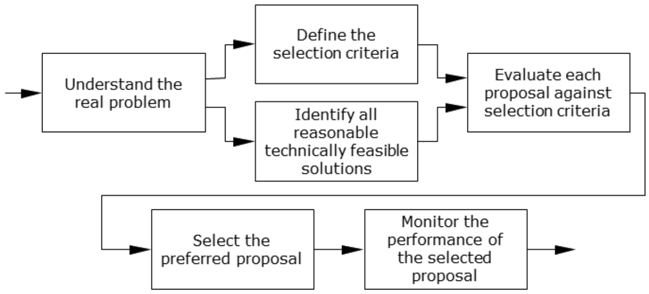

Figure 3: The basic business decision-making process

Figure 3 shows the process as mostly stepwise and serial. The real process is more fluid. Sometimes the steps can be done in different order and often several of the steps can be done in parallel. The important thing is to be sure that none of the steps are skipped or shortcut. It's also important to understand that this same process applies at all levels of decision: from a decision as big as determining whether a software project should be done at all, to a deciding on an algorithm or data structure to use in a software module. The difference is how financially significant the decision is and, therefore, how much effort should be invested in making that decision. The project-level decision is financially significant and probably warrants a relatively high level of effort to make the decision. Selecting an algorithm is often much less financially significant and warrants a much lower level of effort to make the decision, even though the same basic decision-making process is being used.

More often than not, an organization could carry out more than one proposal if it wanted to, and usually there are important relationships between proposals. Maybe Proposal Y can only be carried out if Proposal X is also carried out. Or maybe Proposal P cannot be carried out if Proposal Q is carried out, nor could Q be carried out if P were. Choices are much easier to make when there are mutually exclusive paths—for example, either A or B or C or whatever is chosen. In preparing decisions, it is recommended to turn any given set of proposals, along with their various interrelationships, into a set of mutually exclusive alternatives. The choice can then be made among these alternatives.

1.6. Valuation

In an abstract sense, the decision making process—be it a financial decision or not—is about maximizing value. The alternative that maximizes total value should always be chosen. A basis for comparison based on value is comparing two or more cash flows. Several bases of comparison are available, including:

- Present worth
- Future worth
- Annual equivalent
- Internal rate of return
- (Discounted) Payback period

Due to the time-value of money, two or more cash flows are equivalent only when they equal the same amount of money at a common point in time. Comparing cash flows only makes sense when they are expressed in the same time frame.

Note that value can't always be expressed in terms of money. For example, whether an item is a brand name or not can significantly affect its perceived value. Relevant values that can't be expressed in terms of money still need to be expressed in other terms so that they can be evaluated objectively.

1.7. Inflation

Inflation describes long-term trends in prices. Inflation means that the same things cost more than they did before. If the planning horizon of a business decision is longer than a few years, or if the inflation rate is over a couple of percentage points annually, it can cause noticeable changes in the value of a proposal. The present time value therefore needs to be adjusted for inflation rates.

1.8. Depreciation

Depreciation and amortization are used to handle an asset deduction when you are using accrual accounting for your business. Depreciation is used for tangible expenses, and amortization is used for intangible expenses. In both cases, the cost of assets is spread over the lifetime of a business to provide a more complete view of a business' cash flow and profit. Depreciation thus addresses how investments in capital assets are charged off against income over several years. Depreciation is an important part of after-tax cash flows, which is critical for accurately addressing income taxes. If the software itself is considered a value in itself (unlike, for instance, an embedded component in a larger product), development costs are recorded as capitalized expenditures, which are expenses that have become assets. Since this software product will be sold in more periods than the development costs were incurred, those costs should be capitalized and written off in those subsequent periods. The amount to be depreciated is the software product value minus its estimated residual value, which can be zero. The depreciation expense for each period is the amount to be depreciated divided over the number of periods in which the capitalized expenditure will continue to be of use. Since your software project proposals will be compared against non-software proposals and alternative investment options, you should understand how the non-software proposals are being evaluated and how depreciation relates to initial cost.

1.9. Taxation

Governments charge taxes in order to finance expenses that society needs but that no single organization would invest in. Companies have to pay income taxes, which can take up to 50% of a corporation's gross profit. A decision analysis that doesn't account for taxation can lead to the wrong choice. A proposal with a high pre-tax profit won't look nearly as profitable in post-tax terms. Not accounting for taxation can also lead to unrealistically high expectations about how profitable a proposal is.

1.10. Time-Value of Money

One of the most fundamental concepts in finance—and therefore, in business decisions—is that money has time-value: its value changes over time. A specific amount of money right now almost always has a different value than the same amount of money at some other time. This concept has been around since the earliest recorded human history and is commonly known as "interest." Anyone making a business decision needs to understand interest and how it affects that decision. In order to compare proposals or portfolio elements they are normalized in their cost and value to the net present value.

1.11. Efficiency

Economic efficiency is the relationship between the result achieved (see effectiveness) and the resources used to achieve this result. Efficiency means "doing things right." An efficient behavior, like an effective behavior, delivers results—but keeps the necessary effort to a minimum.

1.12. Effectiveness

Effectiveness is about having impact. It is the relationship between achieved objectives to defined objectives. Effectiveness means "doing the right things." Effectiveness looks only at whether defined objectives are reached—not at how they are reached.

1.13. Productivity

Productivity is the ratio of output over input from an economic perspective. Output is the value delivered. Input covers all resources (e.g., effort) spent to generate the output as well as the influence of environmental factors (e.g., complexity, quality, time, process capability, team distribution, interrupts, feature churn, tools, and language). Productivity combines efficiency and effectiveness from a value-oriented perspective: maximizing productivity is about generating highest value with lowest resource consumption.

2. Life-Cycle Economics

2.1. Product

A product is an economic good (or output) that is created in a process that transforms product factors (or inputs) to an output. When sold, it is characterized by attributes that mean a value to its users. It is a deliverable that creates a value and an experience to its users. A product can be a combination of systems, solutions, materials, and services delivered internally (e.g., in-house IT solution) or externally (e.g., software application) either as is or as a component for another product (e.g., embedded software).

2.2. Project

A project is a temporary endeavor undertaken to create with people a unique product or service. In software engineering, we distinguish different project types (e.g., product development, IT infrastructure, outsourced services, software maintenance, service creation, and so on).

2.3. Program

A program is a set of related projects [1]. Programs are often used to identify and manage different deliveries to a single customer or market over a time horizon of several years.

2.4. Portfolio

A portfolio is the sum of all assets and their relationship to the strategy of the organization and its market position. Portfolios are used to group and then manage simultaneously all assets within a business line or organization. Looking to an entire portfolio makes sure that impacts of decisions are considered, such as resource allocation to a specific project—which means that the same resources are not available for other projects.

2.5. Product Life Cycle

The product life cycle (PLC) refers to the sum of all activities needed to define, develop, implement, build, operate, service, and phase out a product or solution and its related variants. It is subdivided into phases that are separated by dedicated milestones, called decision gates. However, the phases of a software product life cycle are often interleaved and overlapped in various ways. For instance an agile PLC consists of several increments being delivered. With the focus on disciplined gate reviews, the PLC fosters risk management, synchronization with different suppliers, and providing auditable, decision-making information (e.g., complying with product liability needs or governance regulations).

2.6. Project Life Cycle

The project life cycle refers to the set of sequential project phases determined by the control needs of the organizations involved in the project. The project life cycle is typically broken down into five phases—namely, Initiating, Planning, Executing, Monitoring and Controlling, and Closing. The project life cycle and the product life cycle are interdependent, i.e., a product life cycle can consist of several projects and a project can comprise several products.

2.7. Proposals

Making a business decision begins with the notion of a *proposal*. Proposals relate to reaching a business objective—either on the project, product, or portfolio level. A proposal is a single, separate option that is being considered, like carrying out a particular software development project or not. Another proposal could be to enhance an existing software component, and still another might be to redevelop that same software from scratch. Each proposal represents a unit of choice—either you can choose to carry out that proposal or you can choose not to. The whole purpose of business decision-making is to figure out, given the current business circumstances, which proposals should be carried out and which shouldn't.

2.8. Investment Decisions

Investment decisions are made by investors in order to spend money and resources on achieving a target. Investors are either inside (e.g., finance, board) or outside (e.g., banks) the organization. The target relates to some economic criteria, such as achieving a high return on the investment, strengthening the capabilities of the organization, or improving the value of the company.

2.9. Planning Horizon

When an organization chooses to invest in a particular proposal, money gets tied up in that proposal—so-called "frozen assets." The economic impact of frozen assets tends to start high and decreases over time. On the other hand, operating and maintenance costs of elements associated with the proposal tend to start low but increase over time. The total cost of the proposal—that is, owning and operating a product—is the sum of those two costs. Early on, frozen asset costs dominate; later, the operating and maintenance costs dominate. There is a point in time where the sum of the costs is minimized; this is called the minimum cost lifetime.

To properly compare a proposal with a four-year life span to a proposal with a six-year life span, the economic effects of either cutting the six-year proposal by two years or investing the profits from the four-year proposal for another two years need to be addressed. The planning horizon, sometimes known as the study period, is the consistent time frame over which proposals are considered. Effects such as software lifetime will need to be factored into establishing a planning horizon. Once the planning horizon is established, several techniques are available for putting proposals with different life spans into that planning horizon.

2.10. Price and Pricing

A price is what is paid in exchange for a good or service. A price is a fundamental aspect of financial modeling and is one of the four Ps of the marketing mix. The other three Ps are product, promotion, and place. Price is the only revenue-generating element amongst the four Ps; the rest are cost centers.

Pricing is part of finance and marketing. It is the process of determining what a company will receive in exchange for its products. Pricing factors are manufacturing cost, market place, competition, market condition, and quality of product. Pricing applies prices to products and services based on factors such as fixed amount, quantity break, promotion or sales campaign, specific vendor quote, shipment or invoice date, combination of multiple orders, service offerings, and many others. The needs of the consumer can be converted into demand only if the consumer has the willingness and capacity to buy the product. Thus, pricing is very important in marketing.

2.11. Cost and Costing

A cost is the value of money that has been used up to produce something and, hence, is not available for use anymore. In economics, a cost is an alternative that is given up as a result of a decision. A sunk cost is the expenses before a certain time, typically used to abstract decisions from expenses in the past, which can cause emotional hurdles in looking forward. From a traditional economics point of few, sunk costs should not be considered in decision making.

Costing is part of finance and product management. It is the process to determine the cost based on expenses (e.g., production, software engineering, distribution, rework) and on the target cost to be competitive and successful in a market. The target cost can be below the actual estimated cost. The planning and controlling of these costs (called cost management) is important and should always be included in costing.

2.12. Performance Measurement

Performance measurement is the process where an organization establishes the parameters within which programs, investments, and acquisitions are measured to control whether they are reaching the desired results. It means to evaluate whether performance objectives are actually achieved; to control budgets, resources, progress, and decisions; and to learn and improve performance.

2.13. Earned Value Management

EVM is a project management technique for measuring project progress based on created value. At a given moment, the results achieved to date in a project are compared with the projected budget and the planned schedule progress for that date. Progress relates already consumed resources and achieved results at a given point in time with the respective planned values for the same date. It helps to identify possible performance problems at an early stage.

2.14. Termination Decisions

Termination means to end a project or product. Termination can be planned for a long time (e.g., when foreseeing that a product will reach its lifetime) or can come rather spontaneously (e.g., when performance targets are not achieved). In both cases, the decision must be carefully prepared, considering always the alternatives of continuing versus terminating. Costs of different alternatives must be estimated—covering topics such as replacement, information collection, suppliers, alternatives, assets, and utilizing resources for other opportunities. Sunk costs must not be considered in such decision making because they have been spent and will not reappear as a value.

2.15. Replacement and Retirement Decisions

A replacement decision happens when an organization already has a particular asset and they are considering replacing it with something else, like deciding between keeping a legacy software system or redeveloping it from the ground up. Replacement decisions use the same business decision process as described above, but there are additional challenges: sunk cost and salvage value. Retirement decisions are about getting out of an activity altogether, such as when a software company considers not selling a software product any more or a hardware manufacturer thinks about not building and selling a particular model of computer any longer.

3. Risk and Uncertainty

3.1. Goals, Estimates, and Plans

A goal in software engineering economics is mostly a business goal (or business objective). It is external to the specific software engineering project. A goal relates business needs (such as increasing profitability) to investing resources (such as starting a project or launching a product with a given budget, content, and timing). Goals apply to operational planning (for instance, to reach a certain milestone at a given date or to extend testing by some time to achieve a desired quality level) and to the strategic level (such as reaching a certain profitability or market share).

An estimate is the well-founded evaluation of how much resources and time would be necessary to achieve a stated goal (see also “Effort, schedule, and cost estimation” in the Software Engineering Management KA). Estimates are typically internally generated and not necessarily externally visible. They should not be driven by the goals because this could make the estimate overly optimistic. Of course, the underlying solutions driving the estimates should be aligned with the goals.

A plan describes the activities and milestones that are necessary in order to reach the goals of a project (see also “Software Project Planning” in the Software Engineering Management KA). The plan should be in line with the goal and the estimate, which is not necessarily easy and obvious—such as when a software project with given requirements would take longer than the target date foreseen by the client. In such cases, plans demand a review of initial goals as well as estimates and the underlying uncertainties and inaccuracies. Creative solutions with the underlying rationale of achieving a win-win position are applied to resolve conflicts. The plan needs to consider the constraints of the project and achieve commitment with impacted stakeholders to be useful. Figure 4 shows how goals are initially defined. Estimates are done based on the initial goals. The plan finally tries to match the goals and the estimates in order to meet the goals. This is iterative, because typically an initial estimate does not meet the initial goals.

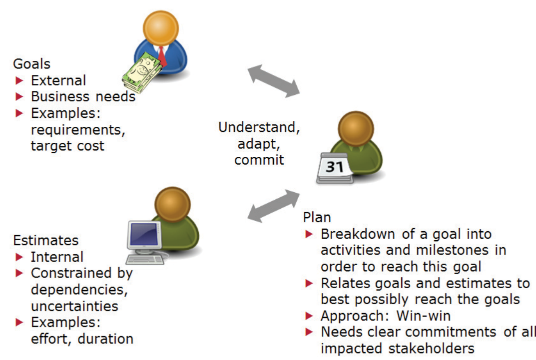

Figure 4: Goals, estimates, and plans

3.2. Estimation Techniques

Estimations are used to analyze and forecast the resources or time necessary to implement requirements (see also “Effort, schedule, and cost estimation” in the Software Engineering Management KA). Four families of estimation techniques exist:

- Expert judgment
- Analogy
- Decomposition
- Statistical (or parametric) methods

3.3. Addressing Uncertainty

Because of the many unknown factors during project initiation and planning, estimates are inherently uncertain; that uncertainty should be addressed in business decisions. Techniques for addressing uncertainty include:

- Consider ranges of estimates
- Analyze sensitivity to changes of assumptions
- Delay final decisions

3.4. Prioritization

Prioritization means to compare alternatives based on different criteria and then rank those alternatives to deliver the best possible value. In software engineering projects, software requirements are often prioritized in order to deliver the most value to the client or to allow for building increments where a first delivery ensures that the client sees a value (see also “Requirements Classification and Requirements Negotiation” in the Software Requirements KA).

3.5. Decisions under Risk

Decisions under risk techniques are used when the decision maker can assign probabilities to the different possible outcomes (see also “Risk Management” in the Software Engineering Management KA). The specific techniques include:

- Expected value decision making
- Expectation variance and decision making
- Monte Carlo analysis
- Decision trees
- Expected value of perfect information

3.6. Decisions under Uncertainty

Decisions under uncertainty techniques are used when the decision maker cannot assign probabilities to the different possible outcomes (see also “Risk Management” in the Software Engineering Management KA). The specific techniques include:

- Laplace Rule
- Maximin Rule
- Maximax Rule
- Hurwicz Rule
- Minimax Regret Rule

4. Economic Analysis Methods

4.1. For-Profit Decision Analysis

Figure 5 describes the process for identifying the best alternative from a set of mutually exclusive alternatives. Decision criteria depend on the business objectives and typically include ROI or ROCE.

The for-profit decision techniques don’t apply when the organization’s goal isn’t profit—which is the case in government and non-profit organizations. In these situations, the organization has a different goal—which means that a different set of decision techniques are needed, such as cost-benefit or cost-effectiveness analysis.

4.2. MARR

The minimum attractive rate of return is the lowest internal rate of return the organization would consider to be a good investment. Generally speaking, it wouldn’t be smart to invest in an activity with a return of 10% when there’s another activity that’s known to return 20%. The MARR is a statement that an organization is confident it can achieve at least that rate of return. The MARR represents the organization’s opportunity cost for investments. By choosing to invest in some activity, the organization is explicitly deciding to not invest that same money somewhere else. If the organization is already confident it can get some known rate of return, other alternatives should be chosen only if their rate of return is at least that high. A simple way to account for that opportunity cost is to use the MARR as the interest rate in business decisions. An alternative’s present worth evaluated at the MARR shows how much more or less (in present-day cash terms) that alternative is worth than investing at the MARR.

4.3. ROI

The return on investment is a measurement of the profitability of a company or business unit. It is defined as the ratio of money gained or lost (whether realized or unrealized) on an investment relative to the amount of money invested.

4.4. ROCE

The return on capital employed is a measurement of the profitability of a company or business unit. It is defined as the ratio of a gross profit before taxes and interest (EBIT) to the total assets minus current liabilities. It describes the return on the used capital.

4.5. Cost-Benefit Analysis

Cost-benefit analysis is one of the most widely used methods for evaluating individual proposals. Any proposal with a benefit-cost ratio of less than 1.0 can usually be rejected without any further analysis because it would cost more than it would benefit. Proposals with a higher ratio need to consider the associated risk of an investment and compare the benefits with the option of taking that same money to the bank.

4.6. Cost-Effectiveness Analysis

Cost-effectiveness analysis shares a lot of the same philosophy and methodology with cost-benefit analysis. There are two versions of cost-effectiveness analysis: the *fixed-cost* version maximizes the benefit given some upper bound on cost; the *fixed-effectiveness* version minimizes the cost needed to achieve a fixed goal.

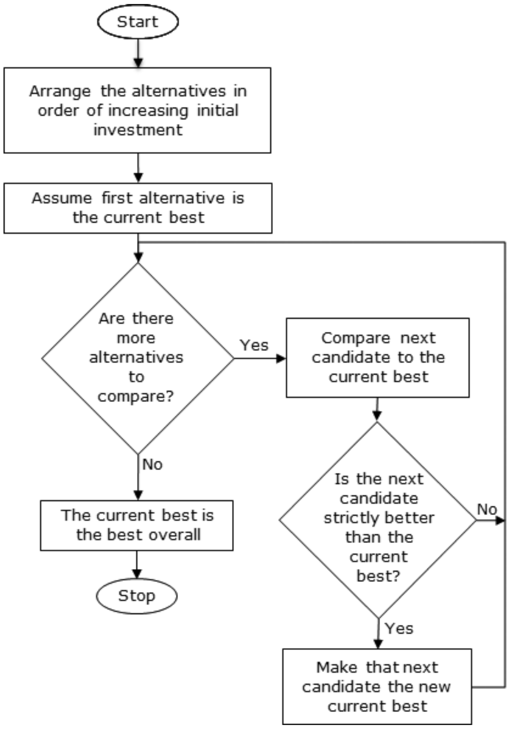

Figure 5: The for-profit decision-making process

4.7. Break-Even Analysis

Break-even analysis identifies the point where the costs and the revenue of a project proposal are equal. Such an analysis can be used to choose between different proposals. Below the break-even point one proposal might be preferred, while above that point another proposal might be preferred.

Given functions describing the costs of two or more proposals, break-even analysis helps in choosing between them by identifying points where the cost are equal.

4.8. Business Case

The business case is the consolidated information summarizing and explaining a business proposal from different perspectives (cost, benefit, risk, and so on) for a decision maker. It is often used for assessing the value of a product or the requirements of a project, both of which can be used as a basis in the investment decision-making process. As opposed to a mere profit-loss calculation, the business case is a "case" of plans and analyses that is owned by the product manager and used in support of achieving the business objectives.

4.9. Multiple Attribute Evaluation

The topics discussed so far are used to make decisions based on a single decision criterion: money. The alternative with the best present worth, the best ROI, etc. is the one selected. Aside from technical feasibility, money is almost always the most important decision criterion, but it's certainly not always the only one. Quite often there are other criteria, other "attributes," that need to be considered, and those attributes can't be cast in terms of money. Multiple attribute decision techniques allow other, non-financial criteria to be factored into the decision.

There are two families of multiple attribute decision techniques that differ in how they use the attributes in the decision. One family is the "compensatory," or single-dimensioned, techniques. This family collapses all of the attributes onto a single figure of merit. The family is called compensatory because, for any given alternative, a lower score in one attribute can be compensated by—or traded off against—a higher score in other attributes. The compensatory techniques include:

- Nondimensional scaling
- Additive Weighting
- Analytic Hierarchy Process (AHP)

In contrast, the other family is the "non-compensatory," or fully dimensioned, techniques. This family does not allow tradeoffs among the attributes. Each attribute is treated as a separate entity in the decision process. The non-compensatory techniques include:

- Dominance
- Satisficing
- Lexicography

4.10. Optimization Analysis

The typical use of optimization analysis is to study a cost function over a range of values to find the point where overall performance is best. Software's classic space-time tradeoff is an example of optimization; an algorithm that runs faster will often use more memory. Optimization balances the value of the faster runtime against the cost of the additional memory.

5. Practical Considerations

5.1. The "Good Enough" Principle

Often software engineering projects and products are not precise about the targets that should be achieved. Requirements are stated, but the marginal value of adding a bit more functionality cannot be measured. The result could be a late delivery or too-high cost. The "good enough" principle relates marginal value to marginal cost and provides guidance to determine criteria when a deliverable is "good enough" to be delivered. These criteria depend on business objectives and on a prioritization of different alternatives, such as ranking software requirements or relating schedule to content and cost. One popular rule towards good enough software is the RACE principle: *Reduce accidents and* *control essence*. Accidents imply unnecessary overheads such as gold-plating and rework due to late defect removal or too many requirements changes. Essence is what customers pay for. Software engineering economics provides the mechanisms to define criteria that determine when a deliverable is “good enough” to be delivered.

5.2. Friction-Free Economy

Economic friction is everything that keeps markets from having perfect competition. It means distance, cost of delivery, restrictive regulations, or imperfect information. In high-friction markets, customers don't have many suppliers from which to choose. Having been in a business for a while or owning a store in a good location determines the economic position. It's hard for new competitors to start business and compete. The marketplace moves slowly and predictably. *Friction-free markets* are just the reverse. New competitors crop up all over, and customers are quick to respond. The marketplace is anything but predictable. Theoretically, software and IT are friction-free. New companies can easily create products and often do so at a much lower cost than do established companies since they need not consider any legacies. Marketing and sales can be done via the Internet and social networks, and basically free distribution mechanisms can enable a ramp up to a global business. Software engineering economics aims at providing foundations to judge how a software business performs and how friction-free a market actually is. For instance, software app competition has recently been severely hampered as such apps must be sold through an app store and comply to that store’s rules.

5.3. Ecosystems

An ecosystem is an environment consisting of all the mutually dependent stakeholders, business units, and companies working in a particular area. In a typical ecosystem, there are producers and consumers, where the consumers add value to the consumed resources. Note that a consumer is not the end user but an organization that uses the product to enhance it. A software ecosystem is, for instance, a supplier of an application working with companies doing the installation and support in different regions. Neither one could exist without the other. Ecosystems can be permanent or temporary. Software engineering economics provides the mechanisms to evaluate alternatives in establishing or extending an ecosystem—for instance, assessing whether to work with a specific distributor or have the distribution done by a company doing service in an area.

5.4. Offshoring and Outsourcing

Offshoring means executing a business activity beyond sales and marketing outside the home country of an enterprise. Enterprises typically either have their offshoring branches in low-cost countries or they ask specialized companies abroad to execute the respective activity. Offshoring should therefore not be confused with outsourcing. Offshoring within the own company is called captive offshoring. Outsourcing is the result-oriented relationship with a supplier, who executes business activities for an enterprise when, traditionally, those activities were executed inside the enterprise. Outsourcing is site-independent. The supplier can reside in the direct neighborhood of the enterprise or offshore, which is outsourced offshoring. Software engineering economics provides the basic criteria and business tools to evaluate different sourcing mechanisms and control their performance. For instance, using an outsourcing supplier for software development and maintenance might reduce the cost per hour of software development but increase the amount of hours and capital expenses due to an increased need for monitoring and communication. (For more information on offshoring and outsourcing, see “Outsourcing” under “Management Issues” in the Software Maintenance KA.)

Matrix of Topics vs. Reference Material

	Tockey, 2005 [2*]	Fairley, 2009 [3*]	Sommerville, 2011 [4*]
1. Software Engineering Economics Fundamentals			
1.1. Finance	c2		
1.2. Accounting	c15		
1.3. Controlling	c15		
1.4. Cash Flow	c3		
1.5. Decision Making Process	c2, c4		
1.6. Valuation	c5, c8		
1.7. Inflation	c13		
1.8. Depreciation	c14		
1.9. Taxation	c16,c17		
1.10. Time-Value of Money	c5,c11		
1.11. Efficiency			c1
1.12. Effectiveness			c1
1.13. Productivity			c23
2. Life-Cycle Economics			
2.1. Product		c6	c22
2.2. Project		c1	c22
2.3. Program			
2.4. Portfolio			
2.5. Product Life Cycle		c2	c2
2.6. Project Life Cycle		c2	c2
2.7. Proposals	c3		
2.8. Investment Decisions	c4		
2.9. Planning Horizon	c11		
2.10. Price and Pricing	c13		
2.11. Cost and Costing	c15		
2.12. Performance Measurement		c7, c8	
2.13. Earned Value Management		c8	
2.14. Termination Decisions	c11, c12		c9
2.15. Replacement and Retirement Decisions	c12		c9
3. Risk and Uncertainty			
3.1. Goals, Estimates, and Plans		c6	
3.2. Estimation Techniques		c6	
3.3. Addressing Uncertainty		c6	

	Tockey, 2005 [2*]	Fairley, 2009 [3*]	Sommerville, 2011 [4*]
3.4. Prioritization		c6	
3.5. Decisions under Risk	c24	c9	
3.6. Decisions under Uncertainty	c25	c9	
4. Economic Analysis Methods			
4.1. For-Profit Decision Analysis	c10		
4.2. MARR	c10		
4.3. ROI	c10		
4.4. ROCE			
4.5. Cost-Benefit Analysis	c18		
4.6. Cost-Effectiveness Analysis	c18		
4.7. Break-Even Analysis	c19		
4.8. Business Case	c3		
4.9. Multiple Attribute Evaluation	c26		
4.10. Optimization Analysis	c20		
5. Practical Considerations			
5.1. The "Good Enough" Principle	c21		
5.2. Friction-Free Economy			
5.3. Ecosystems			
5.4. Offshoring and Outsourcing			

Appendix A. List of Further Readings

B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, Upper Saddle River, 1981. This book is the classic reading on software engineering economics. It provided a first overview on business thinking in software engineering. Although the examples and figures are dated, it still is worth reading.

C. Ebert and R. Dumke, *Software Measurement*, Springer, Heidelberg, New York, 2007. This book provides a profound overview on quantitative methods in software engineering, starting with measurement theory and proceeding to performance management and business decision making.

D.J. Reifer, *Making the Software Business Case: Improvement by the Numbers*, Addison Wesley, 2002. This book is the classic reading on making a business case in the software and IT business. Many useful examples illustrate how the business case is formulated and calculated.