A Preliminary Checklist for Software Cost Management

Magne Jørgensen, and
Kjetil Moløkken
Simula Research Laboratory

Abstract: *This paper presents a process framework and a preliminary checklist for software cost management. While most textbooks and research papers on cost estimation look mainly at the "Estimation" phase, our framework and checklist includes the phases relevant to estimation: "Preparation", "Application", "Cost control", and "Learning". We believe that cost estimation processes and checklists should support these phases to enable high estimation accuracy. The checklist we suggest is based on checklists from a number of sources, e.g., a handbook in forecasting and checklists present in several Norwegian software companies. It needs, however, to be extended through feedback from other researchers and software practitioners. There is also a need for a provision of conditions for meaningful use of the checklist issues and descriptions of the strength and sources of evidence in favor of the checklist issues. The present version of the checklist should therefore be seen as preliminary and we want to get feedback from the conference participants and other readers of this paper for further improvements.*

1 Introduction

This paper suggests a process framework and a preliminary checklist for software cost management based on that framework. The checklist should be tailored to meet a specific project's needs, e.g., reduced to include only the most relevant checklists issues. The goal of the checklist is to increase the cost estimation accuracy, to ensure that work is completed within the approved cost budget and to enable more efficient use of estimation resources. We view software cost management as a sub-discipline of software project management, and software cost estimation as a sub-discipline of software cost management.

The checklist follows the structure of the following software cost management process framework, consisting of four phases and twelve activities:

PREPARATION PHASE
1. Understand estimation problem
2. Agree on decisions and assumptions relevant for estimation
3. Collect information relevant for estimation
4. Select or design the estimation process

ESTIMATION PHASE
5. Estimate most likely cost
6. Assess uncertainty of cost estimate
7. Review estimation process

APPLICATION PHASE
8. Apply estimate in bidding
9. Apply estimate in planning and budgeting
10. Communicate estimate (including communication of uncertainty, bid, plan and budget)
11. Control cost (including re-estimation)

LEARNING PHASE
12. Learn from estimation feedback

This framework is, we believe, valid for most types of software development processes and most estimation work stages, e.g., it should be valid when the estimate is an "early" estimate based on little information and when the estimate is based on a thorough requirement analysis. The importance of each individual checklist issue, however, depends on the type of process and estimate. Although there is a "natural" sequence in the activities described in the framework, most estimation work will be iterative and evolutionary, i.e., there are few intended sequence requirements of the framework.

Most textbooks and papers on software effort estimation discuss issues related to activity 5, i.e., "estimate most likely cost", and a few textbooks and papers discuss activity 6, i.e., "assess uncertainty of cost estimate". We have, however, not been able to find any research paper or software cost estimation textbook that supports the full "cost management life-cycle" as described above.

Some readers may find it surprising that we include activity 11, i.e., "control cost", as an estimation activity. The reason for this inclusion is that management in accordance with the estimate and the uncertainty assessment is needed to achieve accurate estimates. Even the best estimation process cannot lead to accurate estimates if there is lack of control of the resource usage. For example, we have experienced that software work frequently follows the "Parkinson's principle", i.e., that *Work expands so as to fill the time available for its completion* (Parkinson 1947). This means that if a project manager allows a programmer to "gold-plate" his piece of the software product beyond the level of quality expected by the customer, under-estimation is likely. A good estimation process must therefore ensure that there is sufficient management skill and focus on estimation-accuracy-related issues such as efficient use of resources, change request controls, and risk management. Further arguments for this extended estimation context can, among others, be found in software project dynamics textbooks and papers, e.g., (Abdel-Hamid 1990; Abdel-Hamid 1991).

An important assumption leading to the current content of the checklist is that it may be dangerous to view software cost estimation as a rational and mechanical exercise, as many estimation models and process guidelines to some degree seem to. Software cost estimation checklists should, in our opinion, recognize estimation as an activity impacted by "political games" and "tug of wars", not a completely rational process where the only goal of the actors is high estimation accuracy. We have experienced that even in the case of use of "mechanical" estimation aids, such as cost estimation tools, the estimators may strongly bias the outcome, e.g., they may adjust the input values to the estimation tool so that the model's estimated cost fits with what the customer can accept or with the estimators' "gut feeling" of most likely cost. A more extensive discussion on this difference in viewpoints, i.e., the difference between the political and the rational model of viewing software cost estimation, is provided in (Lederer and Prasad 1990). Our viewpoint is also supported by Peter Hall, who in his famous book on "Great Planning Disasters" (Halll 1980), claims that "*... forecasts, often conceived as a mechanical exercise in projecting trends, invariably went wrong for just that reason*".

Note that we do not discard the goal of being rational in the estimation process. What we do is recognize that other goals than estimation accuracy, e.g., the goal of pleasing the manager, frequently dominate the estimation process and that there will be human judgment involved in nearly all estimation processes.

The checklist we suggest in this paper is preliminary. This means that the current version is far from complete and that our goal is to provide a more complete and better structured checklist when we get more feedback. We plan to present the preliminary checklist to several software cost estimation researchers and practitioners to get input and suggestions.

As far as we know, the benefits of checklists have not been empirically validated in software cost estimation contexts. However, there is strong empirical evidence that checklists are helpful in other types of estimation situations. Harvey (2001) provides an overview of forecasting and human judgment studies clearly showing how checklists support people in remembering important variables and possibilities that they would otherwise overlook. Estimators tend to use estimation strategies that require minimal computational effort, at the expense of accuracy, as reported in the time estimation study described in (Josephs and Hahn 1995). Checklists frequently "push" the estimators to use more accurate expert estimation strategies. We have experienced that many software organizations find checklists to be one of their most useful estimation tools. In fact, several software organizations use checklists as their only formal means of "experience database", i.e., the learning from one project leads to changes in the estimation checklist.

2 The Checklist

The current version of the checklist is based on the following sources:

- Estimation checklists from six Norwegian software development companies. (Two of them developed by the author of this paper.)
- Cost management issues in "A guide to the Project Management Body of Knowledge", by the Project Management Institute (2000).
- The expert estimation guidelines in (Jørgensen 2004), (An early version of the paper can be downloaded from: http://www.simula.no/publication_one.php?publication_id=444)
- The checklist "Standards and practices for forecasting" described in (Armstrong 2001).
- Checklists described in software estimation papers, e.g., (Park 1996).
- Checklists described in books on software estimation, e.g., (Boehm, Abts et al. 2000), (Jones 1998) and (Symons 1991).

We suggest the following template for each checklist issue:

1) Description of the checklist issue.

2) Checklist issue rationale.

3) When and how to apply the checklist issue.

4) References to and descriptions of studies and experience relevant for the validation of the checklist issue, e.g., how strong is the evidence in favor of the checklist issue.

For example, the checklist issue 2.3 may be described as this:

<u>Checklist issue 2.3: Consider whether it is meaningful and/or necessary to estimate.</u>

Description: Based on level of estimation problem understanding and completeness of estimation relevant decisions, consider whether it is meaningful and desirable to estimate the software project effort at this stage. Alternatives are, amongst others, to delay the estimation, conduct more extensive pre-studies, or not to estimate at all.

Rationale: There should be consistency between the required estimation accuracy and the means to achieve this accuracy. For example, if there are no important uses of the cost estimates, the effort to derive the cost estimates should reflect this.

Applications: Apply this checklist issue when the organization can influence when to estimate and how much effort to spend on the estimation process.

References: Jørgensen and Sjøberg (2001) report that early estimates easily become "anchors" for the subsequent estimates even in situations where the all stakeholders accept that the early estimate was too low, i.e., early estimates may prevent subsequent accurate estimates. Jeffery and Lawrence (1985) found that the programmers completing tasks without estimates at all worked more efficiently (measured as lines of code per work-hour) than those with estimates. *Note: It is not clear whether the tasks with and without estimates were comparable in that study, e.g., the tasks without estimates may have been easier.*

The use of the checklist could include an assessment with respect to which degree the estimation process of a project follows this issue, e.g., "not relevant/important", "not followed at all (not OK)", "only to some extent followed (partly OK)", and "sufficiently followed (OK)".

This paper does, due to the space limitation of this conference, only include a short description of the checklist issue. An increasingly more complete version of the checklist will be available at our estimation homepage (mail the author for more reference information). It is our intention to update the checklist continuously when receiving feedback from other researchers and practitioners.

The following sections, Section 2.1 - 2.4 contain the checklist issues. Some of the checklist issues may seem to be repeated in different phases. They are, however, different with respect to level of detail due to estimation phase. This will become clearer in the fully explained version of the checklist.

2.1 *Preparation Phase*

1) Understand the Estimation Problem

1.1: Identify goals, stakeholders, deliveries, important assumptions, and main development activities.

1.2: Identify important missing information about the issues in 1.1.

1.3: Agree with the users of the estimate, e.g., the people in charge of a project bid or the project leader, on how the estimate will be used.

1.4: Agree with the users of the estimate about the minimum level of estimation accuracy needed for meaningful use of the estimate, i.e., agree on the importance of accurate estimates.

1.5: Consider how/whether the estimation problem should be divided into sub-problems, e.g., into sub-projects where each sub-project is estimated individually or the estimation work of some sub-projects is postponed.

1.6: Identify "political games" and "tug of wars" that may have impact on the estimation process, i.e., conditions that may hinder a rational estimation process.

2) Agree on Decisions and Assumptions Relevant for Estimation

2.1: Identify decisions and assumptions that have not been made, but having a strong impact on the cost. For example, identify whether it is decided which software professionals and development tools to use or not.

2.2: Make and document the decisions and assumptions necessary for meaningful cost estimation, e.g., it may not be meaningful to continue the estimation process before important technology decisions have been made or a high-level design has been completed.

2.3: Based on level of knowledge and completeness of estimation relevant decisions, consider whether it is meaningful and desirable to estimate the software project effort at this stage of the project. Alternatives are, amongst others, to delay the estimation, conduct more extensive pre-studies, or not to estimate at all.

2.4: Agree with stakeholders on the flexibility and priorities of the project, e.g., whether there is a priority on cost, time-of-delivery, functionality or quality. A software project should be flexible on at least one of these four dimensions, i.e., the customer should, at maximum, set absolute restrictions on the dimensions with priority 1, 2 and 3, and consequently should, at least, allow flexibility on the dimension with priority 4. The only way to meet restrictions of all four dimensions may be to use large contingency buffers.

3) Collect Relevant Information

3.1: Identify and collect information about the most important cost drivers. Use a company-specific checklist to support this process.

3.2: Ensure that the quality and amount of estimation-relevant information matches the estimation situation, e.g., that the completeness and expected stability of requirements matches the need for accurate estimate as input to fixed-price project bids.

3.3: Ensure that the information sources are unbiased, e.g., avoid information being collected by people that are implicitly "rewarded" for low cost estimates.

3.4: Identify biases before applying the data, e.g., identify the reward mechanisms of the people collecting the information.

3.4: Collect information from different sources, e.g., collect the same information from people with different reward mechanisms or roles.

3.5: Collect information about similar software projects completed in own or similar development organization. Do this even if the intention is to apply parametric estimation models in the estimation phase. The information is then useful as an estimation validity check.

3.6: Ensure that the collected information is reliable.

3.7: Avoid the collection of irrelevant and unimportant information. Even when the estimators know that the estimation is irrelevant, irrelevant information may have an unfortunate impact on the cost estimation accuracy.

3.8: Clean the information (e.g., understand the factors that impacted the actual effort of a seemingly similar project, adjust for special events).

4) Select or Design Estimation Process

GENERAL ISSUES

4.1: Base the selection/design of the estimation process on the availability of relevant estimation information and the estimation situation. Do not select/design an estimation model and then search for the, possibly unavailable or low quality, required input information.

4.2: List the available estimation processes and the criteria for selecting/designing them.

4.3: Use organization-specific track-records of the performance of the estimation processes to support the decision. Use person-specific track-records when applying expert judgment-based estimation processes.

4.4: Select/design simple estimation processes unless evidence supports more complex (formal) models.

4.5: Assess the acceptability and understandability of the selected/designed process to its users.

4.6: Be conservative in selection situations with high estimation uncertainty or project instability.

4.7: Follow a written estimation procedure.

4.8: Consider the use of different estimation processes for different parts of the project.

4.9: Combine the use of expert judgment and formal estimation models, when there is uncertainty regarding which will lead to the most accurate estimate.

4.10: Weight the resulting estimates from different processes equally, unless strong evidence supports some of them more than others.

EXPERT JUDGMENT-BASED ESTIMATION PROCESS

4.11: Select experts with good track-record and experience from the estimation of similar projects. Interpret "similar projects" narrowly, e.g., it is not sufficient that an expert has experience with the same technology if the experience is from projects that were much smaller than the current project.

4.12: Select several experts (and combine their estimates) with differences in viewpoints and background.

4.13: Estimate top-down (analogy-based) and bottom-up (activity-based), independently of each other

4.14: Use an extensive, company-specific, WBS (work-breakdown structure) to support the bottom-up estimation process.

4.15: Apply a structured (documented) expert judgment process, not pure "gut-feelings".

FORMAL MODEL-BASED ESTIMATION PROCESS

4.16: Select a model where you believe it is possible to provide high-quality input, especially regarding the expected size of the software product. For example, unusually complex algorithms and interfaces may render lines of code or function point-based size measures less meaningful.

4.17: Tailor the model to your organization. However, data from other, similar, organizations may be used. Do not use un-calibrated models or models developed for different types of projects.

4.18: Update the models frequently.

4.19: Apply only models you understand. Do not use "black-box" models, i.e., models where the tool vendor does not reveal the "inside" of the model.

2.2 *Estimation Phase*

5) Estimate Most Likely Cost (apply the selected/designed estimation process)

5.1: Separate the estimation of most likely cost from the bidding process, e.g., the people in charge of estimating most likely cost should not know about "price-to-win" or other information irrelevant to cost estimation.

5.2: Separate the estimation of most likely cost from the planning process. The project plan has different goals compared with the estimation process, e.g., a project leader may plan an activity with less than most likely use of effort, to put pressure on the developers.

5.3: Present only strongly relevant information to the estimator.

5.4: Avoid estimation situations that favor biases, e.g., when the project leader asks a team member: "You don't need more than 3 work-days on this task, do you?".

5.5: Provide full disclosure of the estimation process.

5.6: Describe the cost estimation assumptions.

5.7: Test as many of the cost estimation assumptions as possible, e.g., that the use of a particular tool should lead to much less use of effort or the assumptions for meaningful use of a formal estimation model.

5.8: If not possible to remove the potential cost estimation biases, identify and describe them.

5.9: Provide easy access to the information used as input to enable replication.

5.10: Ensure that the developers participating in the development have been sufficiently involved and accept the implications of the cost estimates.

5.11: Ensure that there is only one person responsible for the totality of cost estimates and that this responsibility is explicitly defined and documented. In the planning phase, if possible, let the project leader be the one responsible for the cost estimates. This does not however mean that the project leader should estimate himself (this may be a potential bias factor), only that the person in charge of the project is responsible for the accuracy of the cost estimates.

5.12: The cost estimates should take into account the impact from resource and time restrictions, e.g., the higher "speed of delivery", the higher the cost.

5.13: Ask the estimators to justify and criticize their estimates, e.g., try to find reasons why the estimates may be incorrect.

6) Assess Uncertainty

6.1: Identify the main sources of estimation uncertainty, e.g., incompleteness of requirement specification, instable requirements, components developed by others, resource allocation not conducted, and incomplete analysis of implementation consequences.

6.2: Use a company-specific checklist to support the identification of project risks. Update this checklist with new experience after each project.

6.3: Divide uncertainty into four categories: 1) Normal cost variance of activity, 2) Known project risks (positive and negative), 3) Unexpected project events, and, 4) "Crises"/"Chaos". Assess the uncertainty according to uncertainty category. For example: 1) Apply minimum-maximum intervals and cost buffers for normal cost variance of activities, 2) Describe risk + probability of risk + consequences + how to manage the risk for known project risks, 3) Apply contingency buffer for unexpected events, and, 4) Describe actions to re-define the project in case of project crises.

6.4: Use historical data and independent assessors to reduce the typical underestimation of uncertainty.

6.5: Ask the uncertainty assessors to list reasons why the uncertainty assessments may be incorrect.

6.6: Ensure that the expected instability of the requirement is reflected in the uncertainty estimates.

6.7: Assess the acceptable flexibility in the product delivery, e.g., a high flexibility in how the functionality is implemented may decrease the uncertainty significantly.

6.8: Use the historical estimation accuracy of similar projects to support and validate the uncertainty assessment, e.g., if the estimator asses the uncertainty to be much lower than the mean estimation error of similar projects, the estimator should be able to provide a very good argumentation for this reduction in uncertainty.

7) Review Estimation Process

7.1: Use independent reviewers to review the estimation process, the estimate of most likely cost, and the uncertainty assessment.

7.2: Ensure that the review is timely, i.e., that the review results can lead to changes in the cost estimates.

7.3: Use a checklist tailored to the organization’s needs and need for estimation accuracy as checklist for the review, e.g., use a tailored version of the checklist presented in this paper together with checklists of cost factors, risk factors, and work break-down structures to review the estimation process, the cost estimate, and the uncertainty assessment.

7.4: Ensure that the involved people, e.g., the project leader, accept the estimate and the known implications of the estimate.

7.5: Ensure that the use of effort and distribution of effort on activities are consistent with that of previously completed, similar projects.

7.6: Ensure that the estimation process and model have been used properly, i.e., not only that an estimation step has been conducted, but also that is has been conducted properly.

7.7: Ensure that the estimation situations have not been biased by irrelevant information.

7.8: Ensure that the estimators have the necessary experience, training and tools needed for reliable estimating.

7.9: Provide a statement on how reliable the reviewer believes the cost estimates and uncertainty assessment are.

2.3 *Application Phase*

8) Apply Estimate in Bidding

8.1: Use most likely cost, together with the uncertainty information as input. Transform the uncertainty information to a probability distribution and use this probability information to find the lowest acceptable bid.

8.2: Consider the use of risk reducing bids, e.g., bids where the contract includes risk sharing between customer and provider.

<Kitchenham-paper>

9) Apply Estimate in Planning

9.1: Decide on the need for contingency budget and how to manage this buffer based on the uncertainty, divided into different uncertainty types - see 6.4, of the project. For example, the project leader or the steering committee may have separate contingency budgets. A high degree of contingency budget matched with a disciplined process for accessing it may frequently be the main key for project success.

9.2: Plan project activities to reduce the uncertainty as early as possible, e.g., a project may plan the delivery of an early increment involving unfamiliar components and tools to get an early reduction of uncertainty.

9.3: Plan re-estimates to reflect the time-dependent need for cost estimation and time estimation accuracy.

9.4: Do no plan use of over-time. Be especially careful with the planned use of key resources. A most likely cost estimate is of no use if the person supposed to conduct the work is not available or “worn out”.

10) Communicate Estimate, Bid, Plan and Uncertainty

10.1: Identify the maturity of the receivers of the estimation information, e.g., identify the customers’ knowledge about the inherent uncertainty of most software projects. Design the format of the communication based on that customers’ level of software project knowledge and maturity.

10.2: Admit high risk, if high risk is present, but communicate this in a way that is properly understood by the stakeholders.

10.3: Consider provision of alternative, risk-reducing, solutions (and uncertainty estimates), where the alternatives still meet most of the goals of the stakeholders.

10.4: Understand the needs of the receivers of the estimation information, e.g., a budget responsible is not necessarily interested in the dependency between tools, architecture and cost estimates.

10.5: Communicate the estimation information using the language of the receiver.

10.6: Use the “precision” of the cost estimate as indication of the uncertainty, e.g., do not communicate the most likely estimate of cost to be $ 1.456.123 if the uncertainty is high.

10.7: Provide a clear summary of estimates, assumptions made, important implications and information used as input.

10.8: Provide a clear summary of the estimation process.

10.9: Present prediction intervals (minimum-maximum with connected uncertainty levels) to indicate the uncertainty.

10.10: Present important project risks, and if very high risk, suggest alternative solutions and contract types that reduce the risk.

10.11: Communicate the impact of high time-pressure on cost.

10.12: Do not ignore the political behavior of the stakeholders. Know their goals and support their goals, without reducing the realism of the cost estimate.

11) Control Costs (including re-estimation)

11.1: Identify the factors that may lead to lack of cost control, e.g., lack of requirement change control.

11.2: Include management control procedures that prevent incorrect, inappropriate, or unauthorized changes in project scope.

11.3: Re-estimate regularly and when new important information arrives, e.g., changes to requirements that affect cost or schedule, constraints change, and actual values for product or process parameters are found to be significantly different from those on which the estimates are based. If the changes are authorized by the stakeholders, communicate the cost impact and re-plan the project.

11.4: Monitor cost performance to detect and understand variance from plan (performance measurement). Avoid monitoring processes that are misled by the "80% finished, 80% of the time"-trap.

11.5: Experienced developers should supervise the work of inexperienced software developers to avoid inefficient use of resources.

11.6: Apply cost control double-loop learning, i.e., do not only solve cost control problems, understand the reasons for them and prevent the same problems from reoccurring.

11.7: Ensure that all software developers have the "big picture" of the project and how cost (and time) over-runs on one activity may impact other activities and the probability of the success of the project.

11.8: Be especially concerned about the work-hours and work direction of "workaholics", e.g., avoid key personnel getting worn out early in the project and needing to be replaced later on, and work being spent on unimportant issues.

11.9: Reduce the risk of resource conflicts, e.g., if a piece of work can only be completed efficiently by one individual and the cost estimate reflects this, get that work done as soon as possible.

11.10: Keep the designs and technical solutions as simple as possible to reduce the risk, e.g., do not change a design accepted by the customer unless it is flawed or too costly ("the best should not be the enemy of the good").

2.4 *Learning Phase*

12) Learn from Feedback

12.1: Arrange experience reviews (sometimes referred to, morbidly, as post-mortems) at the completion of each project.

12.2: Emphasize cause-effect relationships and "double-loop" learning, i.e., understand the reasons for project events.

12.3: Include a review of when and why the estimates were accurate in this and other similar projects, i.e., do not only analyze failure.

12.4: Consider the need for an update of the company-specific estimation checklist, cost driver checklist, work break-down structure, etc. due to the experience.

12.5: Understand the importance of not generalizing the cause-effects to future projects when there is not sufficient evidence for generalizing. For example, the unexpected cost over-run due to problems with the first time use of development tool X cannot be transferred to future projects if the project members learned much from their experience with that tool.

12.6: Consider the role of luck when learning from project successes and bad luck when learning from project failures, e.g., low cost-overruns may not always be caused by good cost management skills, but can also be a result of luck.

12.7: Let the project members know that the project leader will not accept that possible cost over-runs of individual activities are not communicated as early as possible. Reward early communication of possible cost over-runs.

3 Concluding Remarks

The proposed checklist may, assuming sufficient feedback and investigation of research studies, evolve to a checklist representing state-of-the-art research and best-practice in software cost estimation. In comparison with existing textbooks on software cost estimation and existing checklists, this checklist has a much wider scope. We plan an empirical evaluation of the checklist in Norwegian software development companies through the Norwegian Research Council-supported project SPIKE.

References

2000. A guide to the project management body of knowledge. Pennsylvania, USA, Project Management Institute Inc.

Abdel-Hamid, T. 1990. Investigating the cost/schedule trade-off in software development. IEEE Software(January): 97-105.

Abdel-Hamid, T. 1991. Software project dynamics: An integrated approach. NJ, USA.

Armstrong, J. S. 2001. Standards and practices for forecasting. Principles of forecasting: A handbook for researchers and practitioners. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers: 679-732.

Boehm, B., C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer and B. Steece 2000. Software cost estimation with Cocomo II. New Jersey, Prentice-Hall.

Halll, P. 1980. Great planning disasters, University of California Press.

Harvey, N. 2001. Improving judgment in forecasting. Principles of forecasting: A handbook for researchers and practitioners. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers: 59-80.

Jeffery, D. R. and M. J. Lawrence 1985. Managing Programmer Productivity. Journal of Systems and Software 5(1): 49-58.

Jones, C. T. 1998. Estimating software costs, McGraw-Hill.

Josephs, R. and E. D. Hahn 1995. Bias and accuracy in estimates of task duration. Organizational Behaviour and Human Decision Processes 61(2): 202-213.

Jørgensen, M. 2004. A Review of Studies on Expert Estimation of Software Development Effort. To appear in: Journal of Systems and Software.

Jørgensen, M. and D. I. K. Sjøberg 2001. Impact of software effort estimation on software work. Journal of Information and Software Technology 43: 939-948.

Lederer, A. L. and J. Prasad 1990. Information system cost estimating. MIS Quarterly(June): 159-176.

Park, R. E. 1996. A manager's checklist for validating software cost and schedule estimates. American Programmer 9(6): 30-35.

Parkinson, C. N. 1947. Parkinsons's law or the pursuit of progress. London, John Murray.

Symons, C. 1991. Software sizing and estimating: MkII Function Point Analysis, J. Wiley and Sons.