The Effect of Communication Overhead on Software Maintenance Project Staffing: a Search-Based Approach

Massimiliano Di Penta*, Mark Harman**, Giuliano Antoniol*,***, and Fahim Qureshi**
dipenta@unisannio.it, Mark.Harman@kcl.ac.uk, antoniol@ieee.org, fahim.2.qureshi@kcl.ac.uk

* RCOST — University of Sannio, Via Traiano 82100 Benevento, Italy
**Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK
***Department of Génie Informatique, École Polytechnique de Montréal — Canada

Abstract

Brooks' milestone 'Mythical Man Month' established the observation that there is no simple conversion between people and time in large scale software projects. Communication and training overheads yield a subtle and variable relationship between the person-months required for a project and the number of people needed to complete the task within a given time frame.

This paper formalises several instantiations of Brooks' law and uses these to construct project schedule and staffing instances — using a search-based project staffing and scheduling approach — on data from two large real world maintenance projects. The results reveal the impact of different formulations of Brooks' law on project completion time and on staff distribution across teams, and the influence of other factors such as the presence of dependencies between work packages on the effect of communication overhead.

Keywords: Search-Based Software Engineering, Software Project Management, Software Maintenance.

1 Introduction

Frederick Brooks' book "The Mythical Man-Month" [10], first published in 1975 was one of the first texts to explore the nature of software project management. Brooks was responsible for the development and maintenance of the IBM OS/360 operating system. The Mythical Man Month captured his experiences of software project management. Brooks' work has proved to be highly influential, both within computer since an in the wider world of project management and economics. In 1999 he was awarded the ACM Turing prize for this work.

Brooks' law is often quoted, so much so that it has become something of an aphorism:

> "Adding manpower to a late software project makes it later."

Less widely quoted is the sentence that precedes the statement of Brooks' law in The Mythical Man Month" [10]:

> "Oversimplifying outrageously, we state Brooks' Law"

As this caveat reveals, Brooks was aware that he was simplifying a subtle and complex problem. He was making an observation: there is no simple linear conversion between person–months and staff levels. A project of ten person months' effort, to which two people are assigned will not necessarily take five months to complete.

The manager of a software maintenance project will need to make decisions based on their assessment of the likely impact of Brooks' law. Unfortunately, as stated, the law simply warns against adding people to a late project. This leaves several important questions unanswered: What can a manager do to improve the completion time of a project? How can the impact of the communication overhead be accounted for? What is the impact of the communication overhead on completion time and staffing levels?

This paper aims to explore the relationships between effort, staff levels, completion time and other project factors using various formulations of Brooks' law. The paper formalises these relationships as equations. Previous work on software engineering estimation has focused on fitting equations to existing data to attempt to capture the relationship between effort and completion time [2, 8, 11, 15, 16, 25, 30].

This paper uses search-based optimization techniques — which have recently been applied to software project estimation and planning [4, 5, 7, 12] — to explore the impact of different formulations of Brooks' law upon optimal project allocations, and optimise project attributes such as completion time. The optimization algorithms are applied to data from two large scale real world maintenance projects in order to explore the impact of these different formulations.

This work is valuable in software maintenance where projects may be particularly large and demanding and where there may be existing case histories upon which to draw. Furthermore, the paper shows how search-based staffing can be used to reduce the impact of Brooks' law, balancing large teams — able to quickly maintain large work packages, however suffering more of the communication overhead — with many smaller teams working in parallel with a limited communication overhead.

The primary contributions of this paper can be summarized as follows:

- The paper introduces formal characterizations of Brooks' law.
- The paper introduces a search-based optimization approach to assess the impact of different formulations of the characterizations on real projects.
- The paper presents the results of application of this approach to real world data from two large scale commercial software maintenance projects.
- The paper shows that the impact of Brooks' law is subtle. Tempered by the interplay between the specific values of project attributes, its full effect can only be understood by experimentation and modelling. This motivates the use of the investigative techniques proposed by the paper as a way to explore the impact of Brooks' law on a project-by-project basis.
- The paper presents empirical evidence from two real maintenance projects that suggests that the impact of Brooks' law may be less pernicious for maintenance projects where there is a potential flexibility in team construction.

The remainder of the paper is organised as follows: Section 2 sets out the problems in terms of the Mythical Man Month and introduces our characterizations of Brooks' law. Section 3 explains the search-based approach to optimization of project attributes using the different models of Brooks' law. Section 4 presents the results of an empirical study on the effect of these models of Brooks' law on two large scale maintenance projects. Section 5 discusses the related work, while Section 6 concludes the paper.

2 The Mythical Man-Month

This paper focuses on the problem that people and months are not perfectly interchangeable. However, it should be noted that there are 'perfectly partitionable' tasks in which Brooks' law does not hold, or has minimal effect (see Figure 1-a). Such situations arise where there is an absence of communication and highly standardized tasks are allocated to relatively small teams (up to eight people in one team according to Hamid [1]). For example, the Y2K maintenance project described in reference [3] consisted on the application, through an automatic tool, of a Y2K patch by using a windowing mechanism. This proved to be a highly standardized task with little need for interaction between teams and practically no training and communication overhead.

The opposite case is where there is no possibility to partition the task. No matter how many people are added to the task, the time required will not change (Figure 1-b). Between these two extremes, we have partitionable tasks affected by communication overhead (Figure 1-c). On the one side we have, for each staff member, the need for training, which can be neither partitioned nor squashed. Such effort augmentation factors may be considered to vary linearly with the number of people.

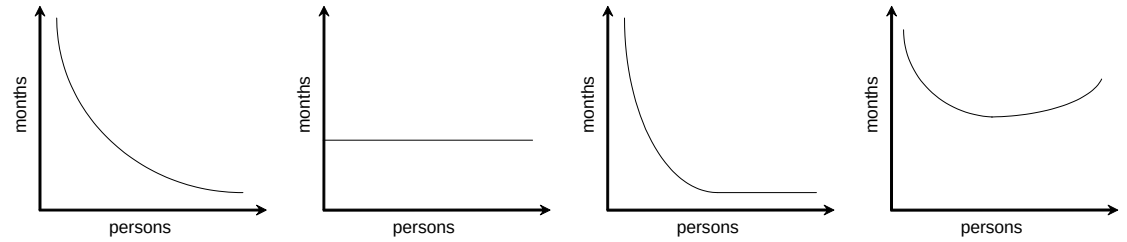
However, there is also a need for intercommunication between different team members. Each member of the n–person team must coordinate her/his activity with $n-1$ other team members. This causes effort increase that can be characterised as some function of $n(n-1)/2$ (the number of pairwise relationships). When relationships are more complex, i.e., there is the need for example for three, four or more people to meet and interact, the situation becomes worse. In some cases, it can happen that the complexity of communication and the effort required to communicate are such that adding further staff to the project results in an increase of completion time (Figure 1-d).

2.1 Characterizing Formal Models of Brooks' Law

The time required for a team staffed with n people to maintain a Work Package (WP) for which the estimated effort is of e people/month will be different depending on the level of communication considered. For a perfectly partitionable task it will be:

$$t = \frac{e}{n} \tag{1}$$

The second case we will consider is the presence of complete communication among team members — thus a pessimistic case of pair communication — limiting however our analysis to pairs and excluding, in the present work,



(a) Perfectly partitionable task (b) Task not partitionable (c) Presence of communication overhead (d) Presence of complex communication overhead

Figure 1. Relationships between people and months [9]

communication among larger number of people. Let ce be the average communication effort for a pair of people. The time needed for communication t_c will be:

$$t_c = ce \cdot N(N-1)/2 \tag{2}$$

thus the time required to perform the task will be:

$$t = \frac{e}{n} + ce \cdot N(N-1)/2 \tag{3}$$

Equation (3) is only one of the possible communication overhead models. As mentioned, in case of complex communication involving groups composed up to k people, t_c will be:

$$\begin{aligned} t_c = \quad & ce \cdot \Big(N(N-1)/2 + N(N-1)(N-2)/3! + \\ & + \ldots + N(N-1) \cdot \ldots \cdot (N-k+1)/k! \Big) = \\ & = ce \cdot \textstyle\sum_{j=1}^{k} \binom{n}{j} \end{aligned} \tag{4}$$

Of course, terms of equation (4) accounting for interactions of j people only apply for teams composed of at least j people (otherwise these terms would bring a negative contribution). In this work we will consider situations for $k = 2$ (i.e., equation (3)) and $k = 3$. Finally, the communication overhead can vary linearly with the number of people composing each team:

$$t_c = ce \cdot N \quad if\ N > 1 \tag{5}$$

or even following a logarithmic model:

$$t_c = ce \cdot log(N) \tag{6}$$

The last two models (for simplicity's sake we will only consider the logarithmic model in our analyses) represent cases where, in presence of large teams, the manager may decide to follow another Brooks' lesson i.e., to structure team communication as to avoid polynomial increase of the communication overhead. Indeed, a logarithmic model mimics the common strategy to organize team members according to some form of network to ensure communication but reduce the related overhead. To the extreme limit of a hierarchical structure, criticized by Brooks, the overhead will be exactly described by a logarithmic function.

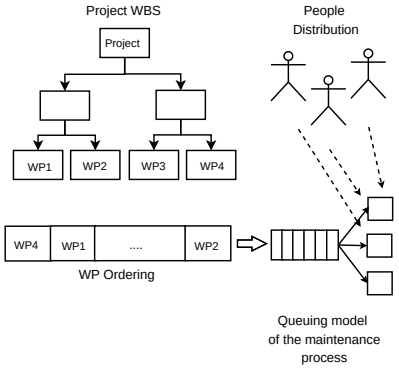


Figure 2. Project scheduling: the queuing model

3 The Search-based Staffing Approach

This section summarises our previous work on search-based optimization for project planning in order to make the paper self–contained. To apply the proposed search-based approach to project planning, the software maintenance process needs to be modelled as a queuing network [3]. In such a model, each stage of the maintenance process is thought of as a queue with a number of servants (i.e., maintenance teams). Different stages are interconnected, forming a queuing system.

Scheduling a software project under the presence of constraints means to determine i) the distribution of the available staff across teams (servants) and ii) the assignment of WPs to teams, with the objective of i) minimizing the

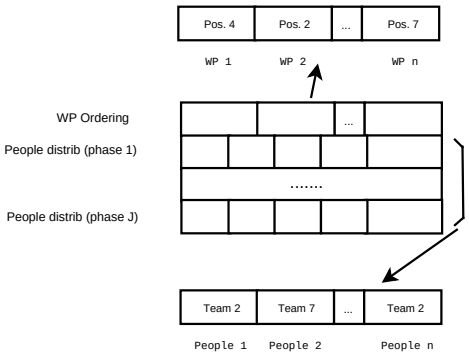


Figure 3. Chromosome representation

project completion time and ii) satisfying precedence constraints between WPs. In addition, the resulting schedule should seek to maximize the team member usage over the project life span; in other words personnel idle periods should be avoided or, at least, limited.

Once the queuing system configuration has been determined (included the size of each server), the WP assignment to teams is defined by the order the WPs enter into the system. Figure 2 depicts the scheduling scenario (i.e., determining WP ordering and people assignment to teams) represented as a queuing-problem. Let us consider, for simplicity's sake, a project modelled as a single-node queuing system. Given this, a solution to the project planning problem can be represented in a 2–array data structure as shown in Figure 3. The first array represents the WP ordering in the incoming queue; it is a N-sized array (where N is the number of WPs), and the value of an entry indicates the position of the WP in the incoming queue, for a single-queue/multi-server queuing system. The second array is an array of size S, where S is the number of people available for the project. Each value of the array indicates the team that a programmer is assigned to. In case of a model composed of multiple (j) queues, the solution will be composed of $1 + j$ arrays, where the first array encodes the WP ordering and the others j encode the people allocation across teams for each maintenance/development phase.

The quality of a solution is quantified as the time to finish, estimated by the queuing simulator. The completion time depends upon the particular queuing configuration (determined by varying the distribution of people across servers) and the given WP ordering. Given the estimated effort for a WP and the staffing level of the team maintaining the WP, the queuing simulator will simulate the WP maintenance determining the working time according to the different models described in Section 2.1.

For many software projects, the search algorithm must be able to handle precedence constraints between WPs. Among the available techniques for handling constraints [13, 26], we choose to repair solutions that yield a good fitness but violate some precedence constraint. Repair attempts to locate a near-neighbour solution that does not violate the constraint in question. Every time a new solution is generated, the process execution is simulated as follows:

1. WPs are seeded into the queue in the order specified by the first line of the chromosome.

2. Every time a server is available, the dispatcher tries to assign it the WP in front of the queue. In case such a WP cannot be handled yet because of a precedence constraint, the schedule searches back in the queue until it finds a WP that can be handled.

3. If the dispatcher is able to find a WP, it will be assigned to the available server. If not, this means that no assignment is currently possible, because one of the WPs currently under work needs to be completed before any of the waiting WPs can be assigned. In that case, simulation proceeds leaving the available server "idle".

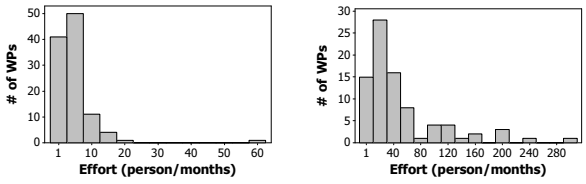
The WP ordering obtained according to the process above described is updated as part of the current solution, so to be used as starting point to generate new solutions (e.g., using crossover and mutation operators when using Genetic Algorithms, as described below).

Once the project planning problem has been modelled, a representation provided and a fitness function described, we can use different search heuristics to solve it. Our previous work showed how Genetic Algorithms (GAs) and Simulated Annealing outperformed other techniques such as Hill Climbing and were in every case significantly better than random search [5]. Building on this finding, we chose to use the GA approach to implement the search based approach reported upon in this paper.

4 Empirical Study

The objective of this study is to analyze the effect of different communication overhead models on the completion time and on the resource assignment to teams when staffing and scheduling a software maintenance project using a search-based approach.

The context of our case studies is two real world maintenance projects, hereby referred as *Project A* and *Project B*. *Project A* is a massive maintenance project concerned with fixing the Y2K problem in a large financial software system from a European financial organisation. According to its Work-Breakdown Structure, the application was decomposed into WPs, i.e., loosely coupled, elementary units (from one to nine for each application) subject to maintenance activities; each WP was managed by a WP leader



(a) Project A (b) Project B

Figure 4. Histograms of WP efforts (person days) for the two projects

and assigned to a maintenance team. No WP dependency was documented and thus no constraint has to be satisfied in *Project A* scheduling. Overall, the entire system was decomposed in 84 WPs, each one composed, on average, of 300 COBOL and JCL files. Further details can be found in reference [3]. *Project A* can be considered as representative of massive maintenance projects related to highly-standardized task such as currency conversion, change of social security numbering, etc.

Project B aimed to deliver the next release of a large data-intensive, multi-platform software system, written in several languages, including DB II, SQL, and .NET. The project is composed of 108 WPs for which WP inter-dependence information is available (in total, there are 102 WP inter-dependencies). In this case, the presence of dependencies between the project's WPs considerably complicates the problems of project management.

Figure 4 shows, for the two projects, histograms of efforts at WP grain-level. For Project A efforts were estimated using using an analogy estimation approach [30], while for Project B we are using actual efforts to analyze how the project completion time could have been changed in presence of communication overhead.

4.1 Research Questions

The research questions this study aims to investigate are the following:

1. **RQ1:** what is the effect of different communication overhead models and levels on the project completion time?

2. **RQ2:** what is the effect of different communication overhead models and levels on the allocation of resources into teams?

3. **RQ3:** what is the effect of different communication overhead models when varying the project staffing?

4.2 Empirical study settings and analysis method

In order to facilitate replication of our work, this section reports the details of the parameter settings used in our simulations.

GA simulations were run considering the following parameters: (i) simple, non-overlapping GA with elitism of two individuals; (ii) population composed of 50 individuals; (iii) 500 generations; (iv) mutation probability 0.1, and crossover probability 0.7. To reduce the bias of randomness, each GA run was repeated 10 times. Though we report plots for average values, differences were always tested using statistical tests, i.e., the Kruskal-Wallis test for multiple means comparison and the Mann-Whitney test for two means comparison. The simulator was implemented in C++, relying on the GA Library *Galib*[1].

We considered three communication overhead models, i.e., a logarithmic model (Equation 6 of Section 2.1), a quadratic model (Equation 3 of Section 2.1), and a cubic model (Equation 4 of Section 2.1, for $k = 3$), a communication effort coefficient ce varying between 0% and 10% of each WP effort. The staffing level considered was the one estimated in reference [3] for Project A (46 people) and the actual one (20 people) for *Project B*. To answer RQ3, these staffing levels were varied between 35 and 55 and between 10 and 35 respectively.

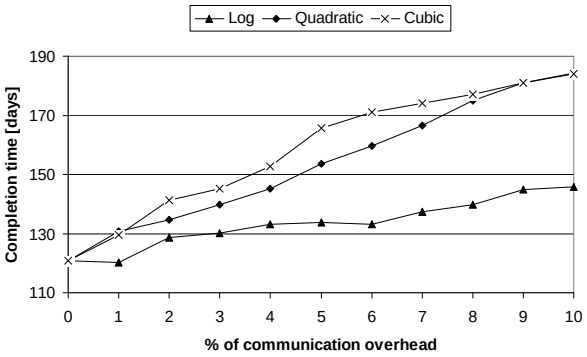
4.3 Empirical Study Results

This section reports results for the simulation of different communication overhead models and levels over data from the two projects, discussing and answering the research questions outlined in Section 4.1.

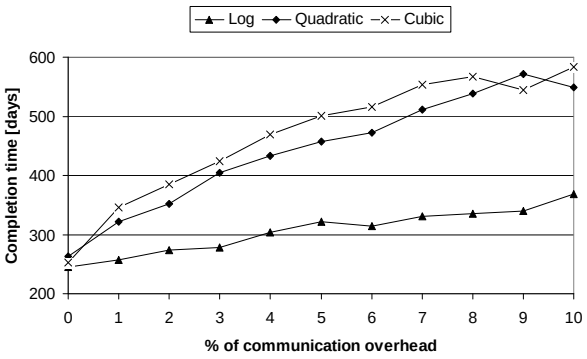
4.3.1 RQ1: what is the effect of different communication overhead models and levels on the project completion time?

Figure 5 shows, for the two projects, the project completion time (averaged over the 10 runs of the GA) for different communication overhead levels and models. For *Project A*, the figure shows how, as the communication overhead increases, the four models behave differently. The Kruskal-Wallis test indicates, for all the percentages of communication overhead, a significant difference (p-values< 0.01). Overall, we found that the logarithmic model always introduces a significantly lower overhead than other models. For a communication overhead greater or equal to 4%, the cubic model introduces a significantly higher overhead than the quadratic model (p-value=0.004). At 9% such a difference is not significant anymore (p-value=0.36), indicating that, as also showed in the figure when the communi-

[1] http://lancet.mit.edu/ga/

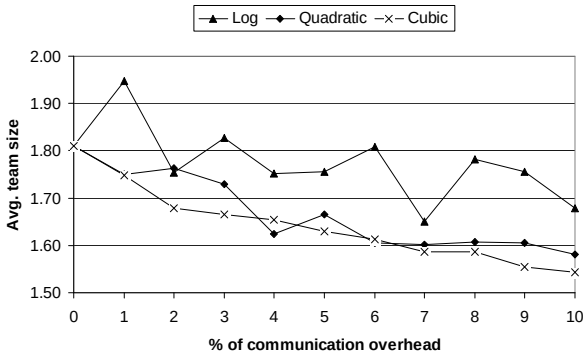


(a) Project A

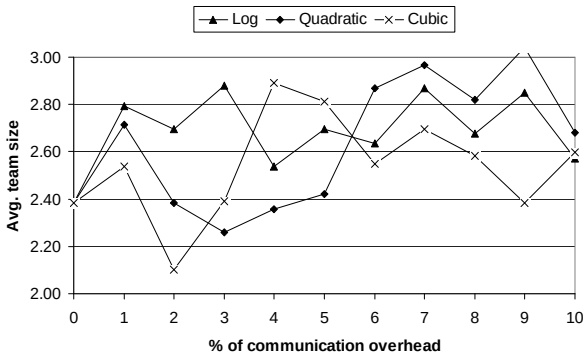


(b) Project B

Figure 5. How completion time varies with communication overhead



(a) Project A



(b) Project B

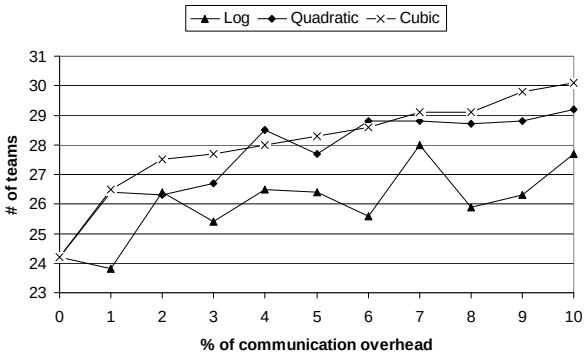
Figure 6. How the average team size varies with communication overhead

cation overhead increases, the cubic and quadratic models tend to converge. Results are confirmed for *Project B*, where, according to the Kruskal-Wallis test, the difference among communication models is significantly different (p-value< 0.0002) mainly because of the smaller communication overhead introduced with the logarithmic model. A detailed analysis shows that the cubic model introduces a significantly higher overhead than the quadratic model, for overhead percentages greater or equal than 2% (p-value=0.008), while the difference is not significant for percentages over 8% (p-value=0.7).

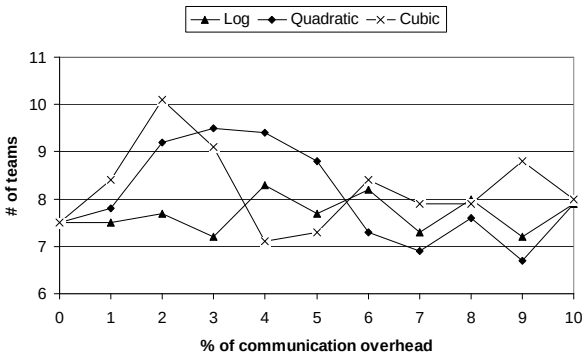
These results are encouraging. By using a re-balancing of project team allocations, the manager is able to find ways of minimizing the impact of communication overheads on the project completion time. This is only possible with a project in which staff can be thought of as equally well skilled at all tasks required. Fortunately, this is true of many project maintenance tasks, including those in which a repeated action is to be applied (such as corrective maintenance).

4.3.2 RQ2: what is the effect of different communication overhead models and levels on the allocation of resources into teams?

For *Project A*, it can be noted how, when the communication overhead increases, the search-based staffing approach tends, on average, to reduce the team size (Figure 6-a), increasing, instead, the number of teams (Figure 7-a). As shown in the figures, this is more evident for the quadratic and cubic models. In other words, instead of preferring large teams — that would have worked better in case of low communication overhead — the approach tends to parallelize the work among many smaller teams, each one affected by a small communication overhead. This also explains why quadratic and cubic models tend to converge in terms of completion time (Figure 5): the search-based approach tend to limit the number of large teams causing differences between the two models. Different models exhibit significant differences in terms of allocated teams (the logarithmic model allocated a significantly smaller number of teams) only for communication overhead percentages be-



(a) Project A



(b) Project B

Figure 7. How the number of teams varies with communication overhead

tween 6% (p-value=0.005) and 8% (p-value=0.002). Also for the average team size, differences are significant between the logarithmic model and other models for percentages of communication overhead above 6%. Table 1 shows an example of staff allocation for *Project A*, considering the cubic model and different percentages of communication overhead. It can be noted how an increase of the communication overhead tends to favour configurations with many smaller teams instead of few larger ones.

For *Project B*, the Kruskal-Wallis test does not reveal any significant difference among the different models both in terms of teams allocated and in terms of average team size. Also in this case, the differences between the two projects can be explained by the presence of WP dependencies in *Project B*. While for *Project A* the search-based scheduling approach was able to mitigate the effect of communication overhead favouring more small teams instead of few large teams, this was not possible for *Project B*, since the work could not be parallelized because of the high number of dependencies. This was visible both in terms of a large effect

Table 1. Project A: example of staff allocation for the cubic model

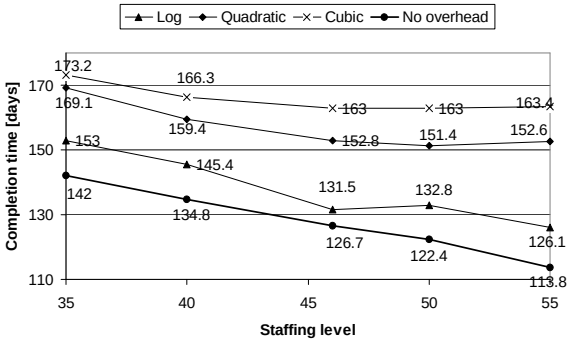
Overhead (%)	Size 1 Teams	Size 2 Teams	Size 3 Teams	Size 4 Teams
0	6	11	6	0
1	18	8	4	0
2	9	14	3	0
3	15	11	3	0
4	19	9	3	0
5	9	14	3	0
6	14	13	2	0
7	17	13	1	0
8	20	11	0	1
9	22	12	0	0
10	24	11	0	0

of the communication overhead on the completion time, but also in terms of a smaller variation of the number of teams and of the average team size.

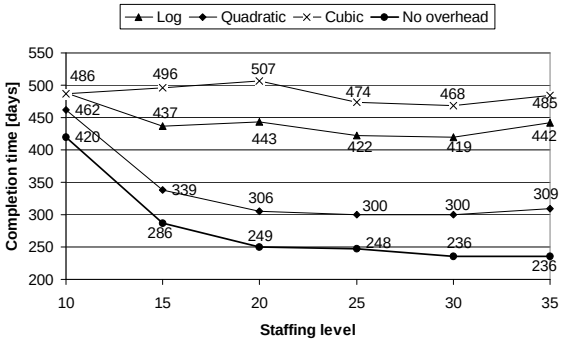
4.3.3 RQ3: what is the effect of different communication overhead models when varying the project staffing?

Figure 8 shows how the completion time varies when the staffing level increases, for different communication overhead models, and for a communication overhead percentage fixed to 5%. The Kruskal-Wallis test indicated a significant difference among different models for all the staffing levels (p-value $< 1.7 \cdot 10^{-7}$). All the models are able to significantly benefit of a staffing increase for staffing levels up to 46 people (p-value=$6.2 \cdot 10^{-5}$ for the cubic model, 0.0001 for the quadratic and for the logarithmic model), although the gain is more evident for the logarithmic model (22 days) than for the quadratic model (16 days) and for the cubic model (10 days). Noticeably 46 was the optimal staffing level determined in reference [3]. Further staffing increases do not cause any significant improvement for the quadratic and cubic model (on the contrary, a slight completion time increase was observed as suggested by the famous Brooks aphorism). The logarithmic model does not benefit from a further staffing increase (up to 50 people). However, when increasing the staff to 55 people, the completion time significantly decreased from 133 to 126 days (p-value=0.05).

This provides further evidence of the way in which the effect of Brooks' law is not uniform and cannot be predicted without recourse to project-specific data. That is, the effect of the law, when applied to a real project is strongly influenced by the particular project data that pertains to the project: the effort required, the staff available, the distribution of tasks to teams in the initial plan, and the WP interdependencies. This observation provides an additional



(a) Project A



(b) Project B

Figure 8. How completion time varies when staffing increases

motivation for the approach adopted in this paper.

For *Project B*, it can be noted that the logarithmic model, and of course the model with absence of overhead, exhibit decreasing completion time when the staffing level increases. When increasing the staffing above 20 people, such an improvement is less evident, and it can also be noted an increase of completion time (although not statistically significant) for the logarithmic model when increasing the staffing from 30 to 35 people. The quadratic model is able to ensure a completion time reduction for a staffing level up to 15 people, then it produces an increase from 437 to 443 days and, subsequently, exhibits oscillations when the staffing level increases. The same happens for the cubic model, which first reacts with a completion time increase for staffing level up to 20 people. Then it is able to reduce the completion time to 468 days (staffing of 30 people) and then starts to increase again for a staffing level of 35 people.

4.4 Threats to Validity

Construct validity threats may be due to the kind of communication overhead models considered. We considered pessimistic cases of communication involving two or three people, while wider communications were not considered also due to the size of teams created by the search-based approach (almost always less than four people). We also considered a logarithmic model, representative of projects with a limited communication overhead effect. Other models will be investigated in future work. Other threats could potentially arise from the simplifications made on the maintenance process modelled (as a single queue model), to the assumptions made in Section 4.2. However, the effect of these threats is limited because the applicability of search-based approaches to deal with project staffing is not influenced by the maintenance process topology. In other words, as discussed in Section 3, the approach would still be applicable if the maintenance process has to be modelled with a more complex queuing network instead of as a single node.

Internal validity threats, in our case study, can be due to the inherent randomness of the search-based algorithms. This was limited by repeating each simulation 10 times and using statistical tests to check for the presence of significant differences.

With regards to *External validity*, threats can be due to the fact that (i) results are limited to data from the two projects and (ii) this is a post-mortem study on real-project data, rather than a case study on running projects. Regarding the fact that this is a post-mortem study, our aim was mainly to analyze the effect of communication overhead and to provide managers with a tool able to model such an effect, rather than empirically assessing the model against actual project data.

5 Related Work

Brooks initiated the study of project planning, following his seminal work on the outcomes of the IBM OS/360 development and maintenance project [10]. Boehm [8] developed a theory of Software Engineering Economics based on fitting models to sets of data from projects on which he had worked in the 1970s and 1980s. This lead him to formulate three models of the relationship between effort and project duration, based on an observed partition of the data into three categories; the organic model (in which Brooks' law has a minimal effect), the semi-detached model (in which Brooks' law applies), and the embedded model (in which Brooks' law is strongly felt).

Boehm's work was widely used in the 1990s by software managers as a prediction mechanism for estimating software effort. Recently other authors have considered the software effort prediction problem, using a variety of techniques including Case Based Reasoning [25, 30, 31], Neural Networks [32] and Search-Based Software Engineering [15, 25]. A recent survey of work on Software Engineering

Economics can be found in a paper by Shepperd [29].

The work in the present paper differs from previous work on estimation of software project effort because we do not aim to fit a predictive model to a set of existing software data. Rather, we seek to experiment with various models of Brooks' law to explore the effect it has on important project attributes, such as staffing levels and completion times. Our approach can be used by a manager to assess the impact of different models of communication on their existing software project estimates and plans.

As such, our work is similar to previous work because it is concerned with providing quantifiable insight into software maintenance economics and decision support to software maintenance engineers and their managers. However, it differs in the approach taken to the provision of this insight and decision support. Our work shows that the effect of Brooks' law is subtle and also potentially less pernicious for maintenance projects compared to other projects.

Search-based techniques have previously been applied to scheduling problems [14, 23]. More recently, several authors have applied search based optimization techniques to software project planning [4, 5, 7, 12] and cost estimation [2, 11, 15, 16, 25]. However, the present paper is the first to provide a formal characterization of models of Brooks' law and to use these to investigate the impact of these models on optimal completion times for real world software maintenance projects.

Search Based Software Engineering (SBSE) techniques are increasingly finding application to problems associated with software maintenance such as modularisation [20, 27], refactoring [22, 28], program comprehension [19], or next release planning [6, 17, 21, 24, 33]. A survey of SBSE can be found in a paper by Harman [18].

6 Conclusion

Communication overhead is a factor that project managers must carefully consider when making their staffing and scheduling decisions. In this paper we analyzed how project completion time and resource allocation varied for different communication overhead models and levels. To this aim, we simulated (near) optimal project manager decisions by relying on a search-based project staffing and scheduling approach. In other words, the search-based approach simulated how the manager could have done her/his best to allocate project resources in presence of communication overhead.

The results show that the effect of Brooks law is subtle, because it is affected by the particular values of project attributes. This makes it important to have a technique to analyse the effect of the law on project completion times and staff allocation plans. This paper provides such an approach. The results are also a source of cautious optimism for the software maintenance community: a proper project scheduling — that can be achieved for instance with the help of search-based optimization techniques — makes the impact of Brooks law less harmful to project completion time.

There are a number of issues to be addressed in future work. First, we need to build empirical evidence of how the different models fit actual project data, in case of projects where the communication overhead was not negligible. Then, it would be useful to consider further communication overhead models, accounting for different development paradigms, e.g., pair programming. Finally, we will investigate the effect of communication overhead on dynamic project restaffing, i.e., on what happens when resources are added during the project as a response to a prediction indicating a possible deadline violation.

7 Acknowledgments

Authors would like to thank Ahmad Faruq for providing data of one of the projects used as case study in this paper. G. Antoniol was partially supported by the Natural Sciences and Engineering Research Council of Canada (Research Chair in Software Evolution #950-202658).

References

[1] T. Abdel-Hamid. The dynamics of software project staffing: a system dynamics based simulation approach. *IEEE Transactions on Software Engineering*, 15(2):109–119, 1989.

[2] J. Aguilar-Ruiz, I. Ramos, J. C. Riquelme, and M. Toro. An evolutionary approach to estimating software development projects. *Information and Software Technology*, 43(14):875–882, Dec. 2001.

[3] G. Antoniol, A. Cimitile, G. A. Di Lucca, and M. Di Penta. Assessing staffing needs for a software maintenance project through queuing simulation. *IEEE Transactions on Software Engineering*, 30(1):43–58, Jan 2004.

[4] G. Antoniol, M. Di Penta, and M. Harman. A robust search–based approach to project management in the presence of abandonment, rework, error and uncertainty. In 10^{th} *International Software Metrics Symposium (METRICS 2004)*, pages 172–183, Los Alamitos, California, USA, Sept. 2004. IEEE Computer Society Press.

[5] G. Antoniol, M. Di Penta, and M. Harman. Search-based techniques applied to optimization of project planning for a massive maintenance project. In 21^{st} *IEEE International Conference on Software Maintenance*, pages 240–249, Los Alamitos, California, USA, 2005. IEEE Computer Society Press.

[6] A. Bagnall, V. Rayward-Smith, and I. Whittley. The next release problem. *Information and Software Technology*, 43(14):883–890, Dec. 2001.

[7] A. Barreto, M. Barros, and C. Werner. Staffing a sowftare project: A constraint satisfaction and optimization based approach. *Computers and Operations Research (COR) focused issue on Search Based Software Engineeering*.

[8] B. W. Boehm. *Software Engineering Economics*. Advances in Computing Science and Technology. Prentice Hall, 1981.

[9] F. Brooks. *The Mythical Man-Month 20th anniversary edition*. Addison-Wesley Publishing Company, Reading, MA, 1995.

[10] F. P. Brooks, Jr. *The Mythical Man Month: Essays on Software Engineering*. Addison-Wesley Publishing Company, Reading , MA , USA, 1975.

[11] C. J. Burgess and M. Lefley. Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, 43(14):863–873, Dec. 2001.

[12] F. Chicano and E. Alba. Management of software projects with gas. In 6^{th} *Metaheuristics International Conference (MIC2005)*, Vienna, Austria, Aug. 2005.

[13] C. A. Coello Coello. Theoretical and numerical constraint-handling tehniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12), January 2002.

[14] L. Davis. Job-shop scheduling with genetic algorithms. In *International Conference on GAs*, pages 136–140. Lawrence Erlbaum, 1985.

[15] J. J. Dolado. A validation of the component-based method for software size estimation. *IEEE Transactions on Software Engineering*, 26(10):1006–1021, 2000.

[16] J. J. Dolado. On the problem of the software cost function. *Information and Software Technology*, 43(1):61–72, Jan. 2001.

[17] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information & Software Technology*, 46(4):243–253, 2004.

[18] M. Harman. The current state and future of search based software engineering. In L. Briand and A. Wolf, editors, *Future of Software Engineering 2007*, Los Alamitos, California, USA, 2007. IEEE Computer Society Press. To appear.

[19] M. Harman. Search based software engineering for program comprehension (keynote). In 15^{th} *International Conference on Program Comprehension (ICPC 07)*, Banff, Canada, 2007. IEEE Computer Society Press. To appear.

[20] M. Harman, R. Hierons, and M. Proctor. A new representation and crossover operator for search-based optimization of software modularization. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1351–1358, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[21] M. Harman, K. Steinhöfel, and A. Skaliotis. Search based approaches to component selection and prioritization for the next release problem. In 22^{nd} *International Conference on Software Maintenance (ICSM 06)*, Philadelphia, Pennsylvania, USA, Sept. 2006.

[22] M. Harman and L. Tratt. Pareto optimal search-based refactoring at the design level. In *GECCO 2007: Proceedings of the* 9^{th} *annual conference on Genetic and evolutionary computation*, London, UK, July 2007. ACM Press. To appear.

[23] E. Hart, D. Corne, and P. Ross. The state of the art in evolutionary scheduling. *Genetic Programming and Evolvable Machines*, 2004.

[24] J. Karlsson, C. Wohlin, and B. Regnell. An evaluation of methods for priorizing software requirements. *Information and Software Technology*, 39:939–947, 1998.

[25] C. Kirsopp, M. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1367–1374, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[26] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics - (2nd edition)*. Springer-Verlag, Berlin, Germany, 2004.

[27] B. S. Mitchell and S. Mancoridis. On the automatic modularization of software systems using the bunch tool. *IEEE Transactions on Software Engineering*, 32(3):193–208, 2006.

[28] M. O'Keeffe and M. O'Cinneide. Search-based software maintenance. In *Conference on Software Maintenance and Reengineering (CSMR'06)*, pages 249–260, Mar. 2006.

[29] M. Shepperd. Software economics. In L. Briand and A. Wolf, editors, *Future of Software Engineering 2007*, Los Alamitos, California, USA, 2007. IEEE Computer Society Press. To appear.

[30] M. J. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743, 1997.

[31] M. J. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In 18^{th} *IEEE International Conference on Software Engineering*, Los Alamitos, California, USA, Mar. 1996. IEEE Computer Society Press.

[32] N. Tadayon. Neural network approach for software cost estimation. In *International Conference on Information Technology: Coding and Computing (ITCC'05) (volume 2)*, pages 815–818. IEEE Computer Society Press, 2005.

[33] Y. Zhang, M. Harman, and A. Mansouri. Multi-objective requirement analysis for the next release problem. In *GECCO 2007: Proceedings of the* 9^{th} *annual conference on Genetic and evolutionary computation*, London, UK, July 2007. ACM Press. To appear.