Software Project Planning for Robustness and Completion Time in the Presence of Uncertainty using Multi Objective Search Based Software Engineering

Stefan Gueorguiev,
Avanade
London Development Centre
135-141 Wardour Street,
London, W1F 0UT
stefan.gueorguiev@avanade.com

Mark Harman,
Department of Computer Science
King's College London
Strand, London WC2R 2LS,
UK
mark.harman@kcl.ac.uk

Giuliano Antoniol,
École Polytechnique de Montréal
P.O. Box 6079, Succ. Centre-Ville,Montréal,
Québec, CA, H3C 3A7
antoniol@ieee.org

ABSTRACT

All large–scale projects contain a degree of risk and uncertainty. Software projects are particularly vulnerable to overruns, due to the this uncertainty and the inherent difficulty of software project cost estimation. In this paper we introduce a search based approach to software project robustness. The approach is to formulate this problem as a multi objective Search Based Software Engineering problem, in which robustness and completion time are treated as two competing objectives. The paper presents the results of the application of this new approach to four large real–world software projects, using two different models of uncertainty.

Categories and Subject Descriptors

D [**Software**]: Miscellaneous; D.2.9 [**Software Engineering**]: Management—*Time estimation*

General Terms

Management, Experimentation, Algorithms

Keywords

Pareto optimality, project planning, software engineering management, Multi objective genetic algorithms

1. INTRODUCTION

Project management is an important part of any software project. In this paper we focus on the software project management activities of defining different tasks, their duration estimates and dependencies and assigning them to suitably qualified teams of staff. In this way a road map is constructed before the project is executed. This road map gives an estimate on the project timeline as well as serving a as guideline on how it is to be executed. The approach presented in this paper is to use SBSE techniques to investigate and control the impact of uncertainty in project planning. We aim to provide the decision makers with a decision support tool that allows them to explore the trade off between earliest completion time and project robustness in the presence of this uncertainty.

It has been known for some time that software project managers find it hard to construct a robust project plan and to manage the risks that might arise [5, 16, 19]. In every commercial project there are unforeseen factors and issues that materialize only after the project is underway. In particularly severe cases, when deadlines are not met or cost becomes too large, these unforeseen factors may lead to premature termination and abandonment of the entire project.

Project managers attempt to anticipate such pitfalls and devise a project plan that can accommodate them, but no one is able to 'foresee the unforeseeable'. Project managers typically construct their plans in such a way that they would accommodate possible pitfalls, without knowing exactly what these pitfalls might be. For example, this can be done by setting certain additional time aside for tackling problems. Of course, such an approach, falling on the side of caution, naturally inflates the overall budget. This conservatism may lead to potentially valuable projects being deemed too risky and/or too costly.

Worse, the manager is only able to be conservative in their plans, they are not able to analyze the trade off between risk and reward. One of the most important rewards for many organisations is early completion. This is particularly important in software projects, where time–to–market is often a paramount concern.

In these situations, a manager may face the invidious task of balancing the risk of project overrun against the risk of over conservatism, leading to a plan that entails an unnecessary lag in time–to–market with consequent loss of market share. What the software project manager needs is an approach to analyze the trade offs between these two objectives. That is, the manager needs to be supported by a tool that allows him or her to explore the space of possible solutions that minimize completion time (ensuring early time to market), while simultaneously building in robustness (so that unforeseen problems do not lead to overrun).

Of course, these two goals of robustness and early completion are in tension; they are naturally contradictory. A highly robust project is naturally one with a lot of built–in flexibility; 'slack' capacity that can be taken up in an emergency. Such a project will not have the earliest completion time. Indeed, the more robust the manager seeks to make their project, the more likely it is that they will delay completion.

However, in any set of project plans, there will exist potential

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

'sweet spots' in the solution space. Despite the general trend that increased robustness results in delayed completion, these sweet spots will be local regions of the solution space, where the manager *can* trade *small* amounts of one objective for a *large* amount of another. This makes the problem ripe for a pareto optimal multi objective search based approach.

This paper introduces such a multi objective solution to robust project planning. It uses the Multi Objective Genetic Algorithm (MOGA) SPEA II to search for the pareto front that denotes optimal choices of robustness and completion time. The paper shows how sweet spots that trade robustness for completion time emerge in real world software project data. The project data used in this paper is drawn from four different real world software projects, drawn from three different software companies, from different locations, including Europe and North America. More specifically, the primary contributions of the paper are as follows:

1. The paper introduces the concept of robustness in project planning using an SBSE analysis, showing how MOGAs can be used to search for optimal solutions that balance the twin objectives of robustness and early completion.
2. The paper presents results that demonstrate that the MOGA is well suited to the problem, applying the 'sanity check' test that it performs convincingly better than a random search allocated the same budget of fitness evaluations.
3. The paper introduces two different models of uncertainty, relating to unexpected additional work packages and to unexpected delay in existing work packages.

The rest of the paper is organised as follows: In Section 2 the robustness/time-to-delivery research problem is defined formally, while Section 3 introduces the search based approach to solving this problem using multi objective pareto optimal search. Section 4 presents the results of the experiments. Section 6 describes the context of related work in which the current paper is located, while Section 7 describes the limitations of the present papers and possible future work. Section 8 concludes.

2. PROBLEM STATEMENT

When talking about project plan robustness we are essentially referring to a property of the plan. We are looking to construct a project plan, given a project definition that consists of a set $WPS = \{wp_1, \ldots, wp_n\}$ of tasks to be performed; a set $P = \{p_1, \ldots, p_m\}$ of available resources/staff; a set $DEPS = \{(wp_i, wp_j)\} : 0 \leq i \leq n$ and $0 \leq j \leq n$ and $j \neq i$ of dependencies between tasks, where wp_j requires wp_i to be completed first and a set $S = \{s_1, \ldots, s_n\}$ of different skills. Each member of the staff in P has a skill associated and each task in WPS requires a certain skill to be performed. For example a task called 'Design Web Front-end' may require an 'HTML skill' that only 3 people in the staff set of 10 may possess.

To solve this optimization problem, we seek an ordering of tasks in the sequence in which they should be completed without violating dependency constraints and an assignment of people to teams. Naturally, for most cases there will be many different ways in which it is possible to arrange the given input.

In our approach we have three objectives to optimize. These are:
Objective O_1: Overall completion time. This is the traditional objective that forms the focus of single objective approaches to the Software Project Planning Problem.
Objective O_2: Completion time difference in the presence of new tasks (a measure of robustness). The objective seeks to reduce project overrun when unexpected new tasks are added. The number of tasks to be added is pre-defined by an error level: $X\%$. In the case of adding WPs, we randomly create $X\%$ more tasks. We then pick a random duration ranging from the minimum duration of all the existing tasks to their overall maximum duration, with uniform probability distribution. Next, we pick a skill, once again, randomly (from the ones already existing in the project definition). Finally, we insert the newly created task at a random position in the processing queue. This insertion simulates dependencies for the new task, because it implicitly requires all these tasks that occur before it in the queue to be completed ahead of it. Of course, the change is made purely to evaluate the fitness in the presence of this uncertainty. When fitness evaluation is complete changes in work packages are undone.
Objective O_3: Completion time difference when tasks' durations are inflated. This is also governed by an error level: $Y\%$. This means that we have picked $Y\%$ of the tasks at random and inflated their required time by a randomly chosen value that ranges from 1 person-hour up to double the original duration of the work package (using a uniform error distribution). It should be noted that all the random choices are generated every time when we evaluate the fitness function. This means that at every evaluation, freshly chosen random tasks will become inflated. As with evaluating O_2 all random choices are generated every time an evaluation is needed.

Therefore, each solution has associated with it an optimisation vector of size $3 : < O_1, O_2, O_3 >$. This vector represents the fitness value of the solution. Due to the definitions of each objective we will always seek to minimise each vector. Therefore, solutions with smaller vectors correspond to fitter solutions that will yield closer-to optimal results. To achieve this we have used a MOGA. How we obtain these values is described in the following section.

3. THE SOLUTION APPROACH

A solution for the scheduling problem is represented by the order in which tasks are processed by teams, the assignment of people to teams and tasks to teams. If dependency constraints exist between tasks the task processing must respect constraints; violating dependency constraints invalidates the scheduling.

It should be noted that for this paper as in previous works [1, 9, 15] we have disregarded Brook's law. That means that the amount of time to complete a task is $\frac{t_{req}}{n}$, where t_{req} is the estimated person-hours required to complete the task and n is the number of people in the team that is assigned to it. Similarly of [1, 9, 15] we also assume that all tasks form a set of non-overlapping activities with possible dependency constraints between them. Each task may also have a skill requirement which means that the task can be completed only by a set of people that have the needed skill. In addition, all available staff is distributed into teams consisting of one or more people. Every person has a skill as well. It should be noted that people with different skills can be in the same team. However when this team is assigned to a task only the people with the required skill can do work while the rest stay idle. This is an impractical solution and, indeed, such solutions are quickly discarded by our search algorithm.

We ground our problem and chromosome representation as well as multi-objective function evaluation on previous works in particular on the ordering genome [1] and queuing simulation [9]. An ordering genome consists of two arrays. The first one represents the task ordering, the order in which tasks are assigned to the first available team. In other words the order in which tasks arrive in the incoming queue of the queuing simulator as in [9]. The second one represents the assignment of people to teams and thus the number of teams or equivalently the number of servants [15].

The crossover and mutation operator for the staffing array are fairly simple. For crossover we use a single-point crossover and the mutation operator simply picks at random two positions in the array and exchanges their teams.

	Project A	Project B	SmartPrice	Database
Hours	4,287	594	1,569	5,390
WPs	84	120	79	115
Staff	20	20	14	7
Skills	1	1	7	5
Deps	No	Yes	Yes	Yes

Table 1: Table of available projects.

The task array manipulation is more complicated. The mutation operator randomly picks two tasks and exchange their position in the queue. The crossover operator is a variation of the single-point crossover, however it was amended to make sure that tasks are not duplicated.

More details on ordering genome, mutation and crossover and how task dependencies are handled, and effects of Brook's law can be found in [1, 2, 15, 14]

3.1 The SPEA II settings used

The algorithm used is the algorithm SPEA II [21]. To cater for random effects, the algorithm was run 30 times and the best pareto front produced was considered. Random search was allocated the same number of fitness evaluations as the SPEA II for comparison. The parameter settings were as follows: Elitism of 10, Population size of 250, 180 Generations for Project A and B, 250 generations for DataBase and SmartPrice. The number of generations was chosen to allow for stabilization with random search allocated the same number of fitness evaluations in each case. Mutation probability was set to 0.1. Crossover probability was set to 0.8. Overall, problem resolution parameters were selected via trial-and-error.

4. EXPERIMENTAL SETUP

The empirical study consists of four real-world project plans. Project A comes from a European financial organisation, Project B comes from a software house. Projects 'SmartPrice' and 'DataBase' come from a large North American corporation with 51 branches throughout North America. Statistics for each project can be found in Table 1.

Project A is a massive maintenance project concerned with fixing the Y2K problem. All tasks in this project are routine ones and require no special skill or any order of completion. 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. SmartPrice is a customer-facing enhancement to the sales process of a sales organisation. This feature provided a more adequate pricing mechanism as well as a method for discounts, voucher use and price conversion. The enhancement concerned has a potentially significant influence on the organisation's revenue stream, so extensive QA was involved. This project involved the web portion of the company's infrastructure with smaller impact on the underlying database and other internal software. The project concluded with an employee training phase. Database is a large scale database upgrade, migrating old (but crucial) Oracle–forms–based system to the newest version of Oracle. The information that was migrated had an estimated value to the organisation of several million dollars and formed the cornerstone of the organisation's operations. About half of the project involved taking precautions against possible causes of data loss. This project primarily involved the Database Administration section of the organisation. However, the Software Application Development section was also involved at the end for training and for upgrading the existing scripts and triggers to make use of the newly available data base functionality.

The research questions that the empirical study aims to investigate are as follows

RQ1: Is our approach better than random search?
Since this is the first attempt to apply SBSE techniques to software project robustness, a natural first question is whether SBSE techniques can out–perform a random search.

RQ2: What insights into trade offs between objectives can be found in the pareto fronts of real software projects?
At the heart of our approach is the concept of pareto optimality as a technique for exploring the inherent tension between the goal of maximizing robustness while minimizing project completion time. This research question seeks to explore the degree to which the pareto fronts for real projects exhibit interesting 'knee points', where a significant improvement in one objective can be obtained at relatively little cost to the other.

RQ3: What is the difference between the two models of uncertainty?
Objectives O_2 and O_3 represent robustness in the presence of two different kinds of uncertainty. This research question asks whether a given project exhibits a similar pattern of trade offs between completion time and these two objectives. That is: does the problem change when new tasks are introduced, compared to when task durations are merely inflated or deflated. At first glance, the addition of a new tasks might seem to be a special case of the application of inflation to an existing task duration. That is, one might ask: 'surely we can simply treat the new task as the inflation of the duration of an existing task that precedes it'. However, this intuition is subtly misplaced. A new task might require different skills and, even if not, it may be attacked by an unused team, whereas an existing task, if extended, cannot be split in this manner. This research question asks whether this subtle difference has any noticable effects on the results.

5. RESULTS AND ANALYSIS

Since there are three objectives O_1, O_2 and O_3, the Pareto fronts are in 3 dimensions. For simplicity and ease of visual reference and comparison we also have projected each Pareto front onto two dimensions. The overall completion time (objective one) always remains on the X axis. For the Y axis we separately project completion time difference when adding tasks and when inflating tasks. That is, the Y axis always represents robustness (lower values indicate less change vulnerability and, therefore, higher robustness), just our interpretation changes.

The projection procedure from three to two dimensions seeks to preserve as much information as possible about the two projected dimensions in the projected pareto front. We first of all project out all values of the 3rd dimension. The result is possibly many value pairs in two dimensions that dominate one another. Out of these 2 dimensional pairs we form the new 2 dimensional pareto front. The points on the 2 dimensional front each denote the best achievable pareto optimal value achievable in two dimensions.

For example, suppose we have the following 3 dimensional triples: $(1,2,3)$, $(3,2,1)$ and $(2,3,1)$ and we want to project onto the first two dimensions. Simply removing the third element of each triple forms the following three pairs: $(1,2)$, $(3,2)$ and $(2,3)$. In the resulting set of pairs the value $(1,2)$ is dominated by one of the other elements (in this case by both) so $(1,2)$ is removed from the set of pairs to form the resulting 2 dimensional pareto front: $(2,3)$ and $(3,2)$.

The alternative projection strategy would be to fix a value for the 'projected out' third dimension and hold it constant, simply select-

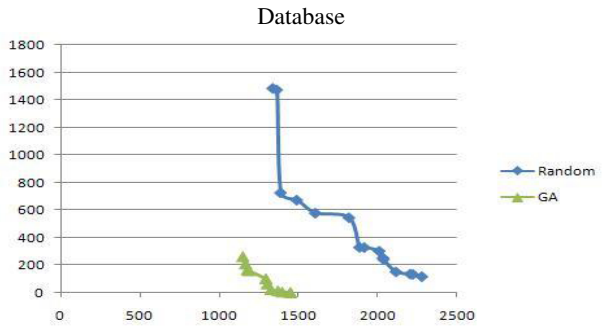

Figure 1: Random Search comparison for different projects under the robustness model in which new tasks are added. That is, the X axis denotes additional units of time (in person hours) required for completion of the tasks if new tasks should be added (objective O_3, while the Y axis represents the total units of time (in person hours) required for the project plan if everything goes as expected (i.e. the earliest completion time; objective O_1). Space restrictions allow sonly for results to be shown for the largest project (Database). Results for the other three projcts are similar.

ing out the corresponding 2 dimensional values for this constant third parameter. However, this forces a somewhat arbitrary choice of fixed constant. It also has the disadvantage that it gives a suboptimal choice of values for the two projected out dimensions.

5.1 RQ1: GA vs Random Search Comparison

Figure 1 shows the comparison between the Pareto fronts produced by our GA algorithm versus a random search implementation for the situation where novel tasks may be added. As can be seen the pareto front for SPEA II dominates the points on the randomly produced pareto front in all four cases. Similar results are obtained from the study of the other possible choice of uncertainty; task inflation.

5.2 RQ2: Reading the Pareto fronts

The results for RQ2 are found by inspecting the pareto fronts for the four real–world projects. These are presented in Figure 2.

In this section we answer RQ2 by considering some of the examples of the fronts, explaining how they reveal insights into the trade off between robustness and project completion time that the manager can exploit in balancing their choices when making decisions about the chosen project plan.

First, consider the first graph of Project B in Figure 2. Here we can see a Pareto front with no knee points. Each point represents a different project plan constructed from the project definition of Project B. The lines connecting the points have been added merely as an aid to the eye in visualizing the front. We can clearly see the trade offs between picking the different plans.

For example, consider the right hand pareto front for the smallest of the projects: Project B. This is the pareto front for the situation in which tasks may become inflated as the project proceeds. From these results, the project manager can be reasonably certain that the project can be completed in 35 person hours if things go as expected (this is a small highly parallelizable project). However, if during execution of the plan, about 30% more tasks are added to the project we can expect to go over our deadline by about one day (6 hours, or 20% of the best achievable completion time).

If we want the project to be robust to these unforeseen additions then we need to budget the extra 6 days potential overrun. In this case there seems to be no way to re-organise the project plan to cater for possible new tasks without simply factoring in the overrun that such additional tasks might yield.

The results for Project B are simple and relatively uninteresting. However, Project B is both very small in overall time required and also its dependencies make the project highly parallelizable, with the consequence that many of the project's 594 total person hours can be allocated in parallel, allowing the project to complete in a mere one week duration. Such very simple projects are unlikely to benefit significantly from an SBSE approach. Nonetheless, as the foregoing discussion illustrates, even for such simple projects, there is a potential for search to reveal some basic relationships between robustness and completion time that may add value for the decision maker.

Now consider the results for Project A in the case where tasks may be inflated during the progress of the project. This situation is captured by the right hand pareto front for Project A in Figure 2. In this case, the pareto front is not linear as it was for Project B. In project A, slight increases in expected completion time can yield a significant increase in robustness. That is, the best completion time available is 220 hours. However, to aim for such an optimal completion time in the project plan leads to a 'fragile' plan; in the presence of novel work packages, the overrun might be as much as 52 hours. However, by delaying expected completion by only 5 hours to 225 hours, we obtain a far more robust plan that is much less fragile. The exposure to overrun is reduced from 52 hours to only 17. The initial part of the pareto front reveals this dramatic trade off in completion time for robustness. The middle section of the pareto front reveals a completely different trade off. That is, in order to achieve even a tiny improvement in robustness, it will be necessary to increase expected completion time from 225 hours to almost 250. This trade off is denoted by the almost flat from 225 to 250 on the X axis. In this way, the pareto front obtained from the estimates of work packages and their interactions for Project A has the potential to give significant insight to the project manager. The manager knows that there would be little point in targeting a completion time between 225 and 250 hours since this cannot allow much increase in robustness. However, the manager would be wise to consider relaxing the best possible completion time from 220 hours to 225, because this would give a greater degree of robustness. Finally, as the last portion of the pareto front reveals, the cost of a really robust project would be a planned–for delay of about 35 hours to 255.

The largest project is 'DataBase'; this project lasts about one year and involves about 3.5 person years' worth of total effort. It also reveals some interesting insights. For instance, if the manager believes that novel tasks are likely to arrive then there is an increased robustness that can be gained for a very small increase in expected completion time. This is indicated by the rapid drop between the first two data points on the left hand pareto front for DataBase in Figure 2. However, attempts to further improve robustness come at a high cost in completion time. Though the pareto front shows some interesting knee points, the overall trade off in increased robustness can only be achieved by building in sufficient 'project slack' to cater for the full expected overrun. Furthermore, the 'long flat tail' of the pareto front indicates that the last small improvement in robustness can only be achieved by building in an enormous amount of project slack. The pareto front allows the manager to asses the variability in likely result and to alert the stake holders in the project to the high degree of uncertainty with which all involved must be prepared to live. If the manager turns to consider the situation in which tasks may become inflated, a similar picture emerges; there is a large increase in robustness that can be

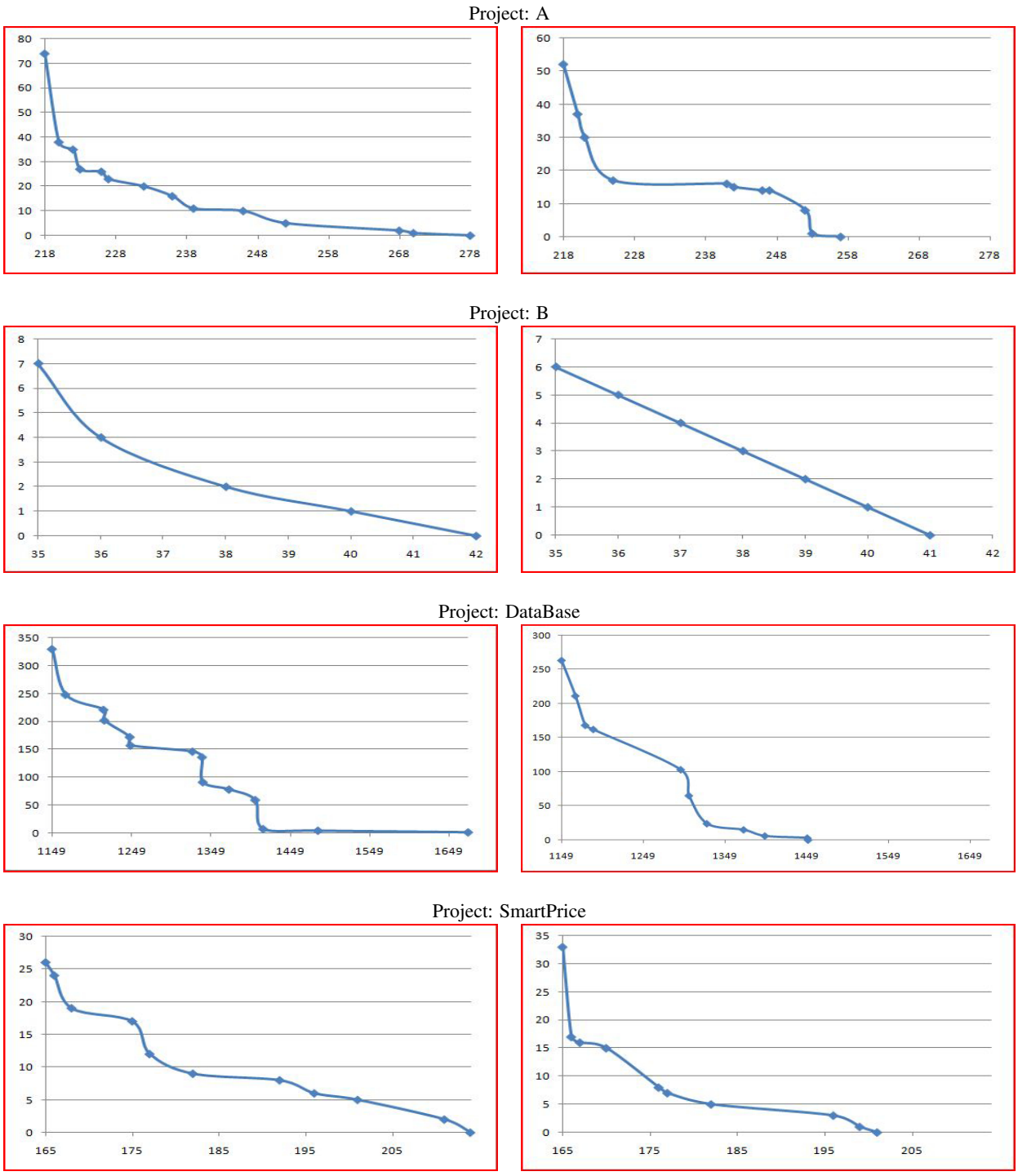

Figure 2: Inflaton of existing tasks vs. addition of unexpected new tasks. The left column shows the results of pareto fronts found for the four projects studied in the situation in which tasks may be added after the project commences. The right hand column presents the results of pareto fronts for the four projects, in the situation in which tasks may become inflated as the project proceeds.

obtained from a little re-planning that involves only a small delay to the earliest possible completion time. However, further robustness increases are unlikely to be rewarding and the final reduction in robustness to eliminate risk due to this uncertainty comes at far too high a price in completion time. That is, removing the risk of a 275 hour overrun can only be achieved by budgeting for a 300 hour delay in completion (from 1149 hours to 1449 hours).

This last observation may initially seem counter intuitive. How can we need 300 hours to increase slack in order to cater for only 250 hours of possible overrun due to uncertainty? Surely we need no more than a *maximum* of x hours delay in expected completion time in order to avoid an overrun of x hours? The answer to this question illustrates the extremely high price of full robustness. To aim for full robustness, we have to have teams of suitably qualified staff available to cater for *any* unexpected event(s). In the specific example of the database project, this does not mean simply allowing one team to be free for 250 hours at the end of the project to pick up the maximum likely overrun; we have to able to handle expected inflation of any tasks at any point. This might come from several different tasks that might be inflected by a small amount each, with et overall effect of changing the critical path of the project dramatically. Alternatively, perhaps one task on the critical path is inflated by a large amount.

Notice that in seeking robustness, we are not designing a project plan to handle a particular unexpected event; we are designing a single project plan that is capable of dealing with a range of unexpected events. This is why the cost of avoiding *any* overrun from 1 to x hours may cost more than x hours in completion time; indeed it usually will in all but the simplest projects.

Returning to RQ2, the results indicate that these 'knee points' are seen in several of the pareto fronts, both for the case where new tasks may be added to the project and also the case in which tasks become inflated during the progress of the project. Since these are real software projects, there is some encouraging evidence to believe that useful insight can be gained from the analysis of pareto fronts, thereby answering RQ2. In each case, these knee points provide important insights for the decision maker. They allow the project manager to make informed decisions about the likely 'sweet spots' where a degree of valuable additional robustness can be achieved at little cost to the overall completion time. Conversely, they also show situations in which a less risk averse manager could choose to trade some robustness for a large reduction in the expected duration of the project.

Notice that these graphs provide useful insight to a project manager who exhibits *any* and *every* form of decision making predilection; the pareto fronts are equally useful to risk–averse managers as they are to managers who seek to take bold risks. Whatever the decision maker's temperament, these pareto fronts can be used to support their decision making and to inform them of the potential consequences of their decisions.

5.3 RQ3: Inflating existing tasks vs. adding new tasks

In the final research question considered in this paper, we explore the difference between the two primary ways in which a project can suffer 'unexpected events'. That is, the project may have task durations that become inflated. Inflation is quite common in software projects, due to the inherent difficulties in estimation and the unavoidable tendency to estimate zero duration for any unexpected activities [16].

Perhaps it is less likely that a project will suffer task addition than it is that tasks will be inflated, but this other form of uncertainty remains a concern for many project managers and cannot be ignored. For this reason we chose to study these two types of uncertainty in isolation to explore the question of whether their impact on a project is different.

From the graphs in Figure 2, it can be seen that the projects do, indeed, react in different ways to these two different forms of uncertainty. For instance, there is a pronounced knee point in the pareto front for Project A for the situation where tasks may become inflated, but this knee point is far less evident in the pareto front for the same project under the uncertainty that tasks may be added.

Interestingly, despite these clear differences, there does appear to be some tantalising evidence to suggest that projects have what might be termed an 'inherent uncertainty profile' in the presence of both types of uncertainty. That is, Project B exhibits a linear pareto front, while the Database and SmartPrice projects show a similar pattern in the pareto fronts for each kind of uncertainty. More research is required to consider this question in more detail. There are simply too few projects here to allow us to confidently generalize.

6. RELATED WORK

SBSE has previously been applied successfully to project scheduling by Davis [13]. A survey of the application of GAs to solve scheduling problems has been presented by Hart *et al.* [7]. A general introduction and survey of recent achievements in Search Based Software Engineering can be found in the survey by Harman [10]. The mathematical problem encountered is an instance of the bin packing or shop-bag problem which are known to be NP-hard. A survey of approximated approaches for the bin packing problem is presented by Coffman *et al.* [12]. More recently Falkenauer published a book devoted to the GA and grouping problems [8].

Some of the closest related work to that presented here is due to Chicano and Alba [6], who used a weighted multi objective GA to combine various project attributes into a single objective search. Their results are obtained using synthetic project data, rather than using real project data, so they are able to explore specifically generated instances, but not the behaviour of their approach for real-world scenarios. Their model was also fundamentally different in two ways; they used weighted fitness aggregation rather than pareto optimality and they did not consider the question of robustness.

Recently, multi-objective approaches to find solutions for the next release problem have been proposed [18, 20]. This previous work on requirements has also considered multiple objectives. In the case of Zhang *et al.* [20] the two objectives are cost and value, which are naturally in tension. In the case of Saliu and Ruhe [18] the objectives are conceptual and implementation based, where there is a tension between levels of abstraction. In both cases, the authors adopt a Pareto optimal approach, like that adopted in the present paper, reflecting a more general migration from weighted optimal aggregated fitness functions to more sophisticated multi-objective approaches [10].

The next release problem is related to the project management problem because both concern the early phases of the project lifecycle. Indeed, requirements analysis can be thought of as a precursor to project planning, though perhaps a better approach would be to combine both activities to simultaneously optimize both. In that way we would be able to choose sets of requirements that best suited the customers, developers and manager(s), treating each as an objective. However this synthesis of early lifecyle software engineering optimization problems remains a task for future work.

The present work differs from previous contributions since we are studying a multi-objective problem where the manager is seeking to produce a robust schedule thus minimizing the risks of project delay or failure due to unexpected events. In particular, we model

unexpected events such as adding activities and inflating task efforts.

We ground our multi-objective approach on the use of queuing simulations previously proposed in Antoniol *et al.* [9]) to implement the objective function. In other words, rather than considering a fixed ordering of WPs and a uniform allocation of developers across teams, we use heuristics to determine the developers distribution and the WPs ordering (and thus their allocation to teams), with the (multiple) objective of minimizing completion time, reduce the effect of inflation of activities and the effect of added tasks thus producing Pareto front representative of robust schedules.

Queuing theory was also applied by Ramaswamy [17] to model software maintenance projects. Simulations of a software maintenance process were performed by Podnar and Mikac [11] with the purpose of evaluating different process strategies rather than staffing the system. Recently, Bertolino *et al.* [4] proposed the use a performance engineering technique, based on the use of queuing models and UML performance profiles, to aid project managers for decision making related to the organization of teams and tasks. We share with them the idea of using queuing networks to model software processes and to support managers in their choices. However, our goal is to suggest to managers a robust schedule able to minimize the completion time and the risk of the project to be late due to wrong effort estimates and unexpected activities. Our work can also be viewed as an attempt to implicitly maximize resource usage, also considered an important issue by Bertolino *et al*.

Other than search-based techniques, there are alternative approaches for project scheduling. Recently Barreto *et al.* [3] have applied constraint satisfaction to staff software projects. However, their focus was different from ours: they aimed at assigning maintenance requests to the most qualified team in terms of skills, or to the cheapest team, or to the team having the highest productivity.

7. LIMITATIONS AND FUTURE WORK

This section sets out some of the limitations of the present work and the extent to which we were able to cater for them in our experiments. Some possible directions for future work are also discussed.

In an assessment of software project planning, it is important to be able to assess the impact of chosen solutions on real world software projects. Obtaining data from real world projects is hard, because often this data is not retained and is only constructed in an informal manner. Where the data is achieved, it is often considered to be among the most sensitive data possessed by a company and so there is a natural reticence to make such commercially sensitive data available for academic research.

In the case of the work presented here, we have not been able to reveal the companies involved, nor are we able to make the data available to other researchers. However, to try to provide some degree of replicatability, we have given a description of the projects for which project management data has been obtained. To try to achieve a measure of coverage we have drawn data from three different companies, and for a variety of types of projects. However, with four case studies there is still a degree of caution required in generalizing on the results we have been able to obtain. This caution is reflected in our choice of research questions. The primary research question (RQ2) simply asks whether there *exist* interesting pareto fronts in real world projects. It does not seek to claim that such fronts are always present, nor even that they are likely; there simply is insufficient data to allow such a strong claim.

These observations would typically be characterised as 'external' threats to validity of the findings, because they affect the degree to which one is able to generalize from our findings. There are also threats to the internal validity of the experiments we conducted that must be consider. The primary threat here comes form the inherent stochastic nature of the algorithms we use. The standard approach to overcoming this stochastic nature, is to repeat the application of the algorithms many times before considering results. In our experiments, we repeated each application of SPEA II thirty times, collecting data from all runs and plotting the pareto front from the best of all thirty runs.

However, there is a further way in which the randomness of the algorithms can impact on our solutions. This lies at the heart of our approach. In order to assess fitness, we create uniformly random instances of the occurrence of unexpected random events. In this way, our fitness function is calculated from a very simple form of simulation. This means that different evaluations of fitness for the same individual can yield different results.

Clearly this introduces an extra degree of randomness into the algorithms. Our hope is that this will be stable at the macro level even though it may fluctuate at the micro level. That is, we hope that the overall results will be stable due to the large number of samples of the search space and the repetition of fitness calculation for both individuals and the exponentially large number of schema within them.

In order to check whether this hope was justified, and to explore the impact of this possible internal threat to validity, we repeated the entire experiment for two of the projects. That is, we constructed ten different pareto fronts using the entire overall approach for projects A and B. We plotted these together to assess the degree of variability of the results. The results are shown in Figure 3. As can be seen the pareto fronts are very similar, and so there is reason to believe that stability is reasonable.

Future work will consider alternative ways of assessing fitness. Essentially, we seek a way of assessing, either a worst case or a typical case for the impact of uncertainty on a particular project plan. Unfortunately this is, in itself, an optimization problem. Treating it as one, would potentially lead to a fitness function that was much more stable, but only at the expense of an enormous increase in computational cost for the overall approach.

Another limitation of the work comes form our attempt to explore the effects of three different objectives at once and to give insight to the manager form the pareto fronts obtained. We are not expecting the manager to possess any expertise in optimization algorithms nor Search Based Software Engineering. Therefore, we need a visualization that is intuitive and from which the manager can draw broad conclusions without any need for an understanding of the underlying algorithms. To achieve this, we chose to draw two dimensional projections of the three objectives, since these have a natural intuitive character, showing a tradeoff between two attributes.

We also experimented with ways for drawing all three objectives. These provide interesting pareto planes, which may be useful to researchers in software project management, However, they may not yield such directly exploitable insight for the practicing software project manager. They also do not show the real solution points so clearly. Rather they can only be used to notice general trends. Figure 4 shows the three dimensional pareto fronts we obtained from the unprojected pareto optimal solutions found for the project SmartPrice as an illustration.

8. CONCLUSION

This paper showed how the problem of accommodating degree of robustness within a software project plan can be formulated as a multi objective Search Based Software Engineering problem. The paper introduced two different models of robustness, with respect to uncertainty about work package cost estimates and uncertainty

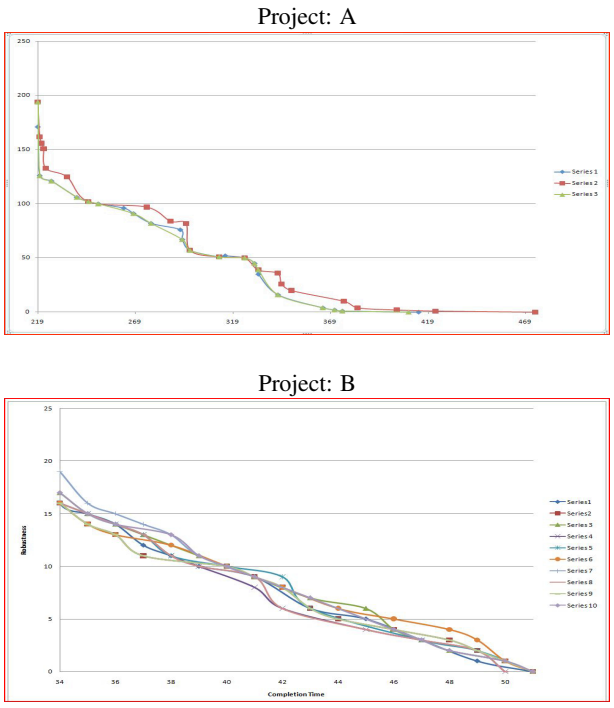

Figure 3: Results of repeating the entire experiment ten times for two of the projects considered. Each pareto front is obtained from 30 runs of SPEA II on the project data. The different pareto fronts give an assessment of the variability of results due to the random nature of the algorithms used and the fitness computation.

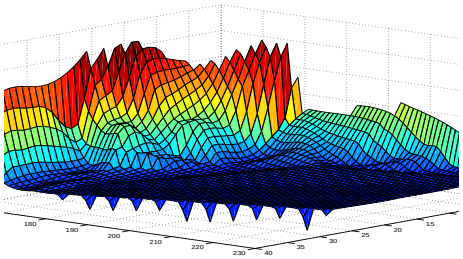

Figure 4: Three dimensional plot of the interpolated pareto surface for all three objectives for the project SmartPrice. The x axis shows (**) the y axis shows (**) and the z axis shows (**). The pareto front is interpolated to connect the 3 dimensional solution points found.

about whether all work packages had been properly accounted for. The paper presented the results of a study of robustness optimization with respect to four real world projects, the results indicate that the search–based approach successfully passes the 'base line sanity check'; it can comfortably out–perform a random search. The results also reveal that real world projects have interesting and potentially insight–yielding pareto fronts and that the two different forms of uncertainty give rise to very different kinds of pareto front (at least in the examples studied herein).

9. REFERENCES

[1] G. Antoniol, M. D. Penta, and M. Harman. Search-based techniques applied to optimization of project plannin for a massive maintenance project. In *Proceedings of IEEE International Conference on Software Maintenance*, pages 240–249, Budapest, Sept 2005. IEEE Computer Society Press.

[2] G. Antoniol, M. D. Penta, M. Harman, and F. Qureshi. The effect of communication overhead on software maintenance project staffing: a search-based approach. In *Proceedings of IEEE International Conference on Software Maintenance*, pages 315–324, Paris FR, Oct. 2-5 2007. IEEE Computer Society Press.

[3] A. Barreto and M. de Barrosand L. Werner. Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers and Operations Research*, 2008 (to appear).

[4] A. Bertolino, E. Marchetti, and R. Mirandola. Performance measures for supporting project manager decisions. *Software Process: Improvement and Practice*, 12(2):141–164, 2007.

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

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

[7] H. E., C. D., and R. P. The state of the art in evolutionary scheduling. *Genetic Programming and Evolvable Machines*, 2004 (to appear).

[8] E. Falkenauer. *Genetic Algorithms and Grouping Problems*. Wiley-Inter Science, Wiley - NY, 1998.

[9] A. G., C. A., D. L. G. A., and D. P. M. Assessing staffing needs for a software maintenance project through queuing simulation. *IEEE Transactions on Software Engineering*, 30(1):43–58, Jan 2004.

[10] 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. This volume.

[11] P. I. and M. B. Software maintenance process analysis using discrete-event simulation. In *European Conference on Software Maintenance and Reengineering*, pages 192–195, Lisbon Portugal, March 2001. IEEE Society Press.

[12] C. E. Jr, G. M.R., and J. D.S. Approximation algorithms for bin-packing. *In Algorithm Design for Computer System Design*, 1984.

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

[14] D. P. M., A. G., and H. M. The use of search-based optimization techniques to plan software projects: an approach and an empirical study. Technical report, RCOST - Univ. of Sannio Italy, 2007. *http://rcost.unisannio.it/mdipenta/searchBasedStaffingTR.pdf.*

[15] M. D. Penta, M. Harman, G. Antoniol, and F. Qureshi. The effect of communication overhead on software maintenance project staffing: a search-based approach. In Proceedings of IEEE International Conference on Software Maintenance*, pages 315–324, Oct 2007.*

[16] R. Pressman. Software Engineering: A Practitioner's Approach*. McGraw-Hill Book Company Europe, Maidenhead, Berkshire, England, UK., 3rd edition, 1992. European adaptation (1994). Adapted by Darrel Ince. ISBN 0-07-707936-1.*

[17] R. Ramaswamy. How to staff business critical maintenance projects. IEEE Software*, 7(7):90–95, May 2000.*

[18] M. O. Saliu and G. Ruhe. Bi-objective release planning for evolving software. In ESEC / SIGSOFT FSE*, pages 105–114, New York NY USA, 2007. ACM Press.*

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

[20] Y. Zhang, M. Harman, and S. A. Mansouri. The multi-objective next release problem. In GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1129–1137, New York NY USA, 2007. ACM Press.*

[21] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov. 1999.*