Multiobjective Simulation Optimisation in Software Project Management

Daniel Rodríguez*
Dept of Computer Science
University of Alcalá
28871 Alcalá de Henares,
Madrid, Spain
daniel.rodriguezg@uah.es

Mercedes Ruiz*
Dept of Computer Science
University of Cádiz
11002 Cádiz, Spain
mercedes.ruiz@uca.es

José C. Riquelme
Dept of Computer Science
University of Seville
41012 Seville, Spain
riquelme@us.es

Rachel Harrison
School of Technology
Oxford Brookes University
Oxford OX33 1HX, UK
rachel.harrison@brookes.ac.uk

ABSTRACT

Traditionally, simulation has been used by project managers in optimising decision making. However, current simulation packages only include simulation optimisation which considers a single objective (or multiple objectives combined into a single fitness function). This paper aims to describe an approach that consists of using multiobjective optimisation techniques via simulation in order to help software project managers find the best values for initial team size and schedule estimates for a given project so that cost, time and productivity are optimised. Using a System Dynamics (SD) simulation model of a software project, the sensitivity of the output variables regarding productivity, cost and schedule using different initial team size and schedule estimations is determined. The generated data is combined with a well-known multiobjective optimisation algorithm, NSGA-II, to find optimal solutions for the output variables. The NSGA-II algorithm was able to quickly converge to a set of optimal solutions composed of multiple and conflicting variables from a medium size software project simulation model. Multiobjective optimisation and SD simulation modeling are complementary techniques that can generate the Pareto front needed by project managers for decision making. Furthermore, visual representations of such solutions are intuitive and can help project managers in their decision making process.

*Part of this work was carried out while visiting Oxford Brookes University

GECCO'11, July 12–16, 2011, Dublin, Ireland.

Track

Search Based Software Engineering

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem solving, control, methods and search—*Heuristic Methods*; I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis; D.2.9 [**Software Engineering**]: Management—*Cost estimation, productivity, time estimation*

General Terms

Management

Keywords

Software Project Management, Simulation Optimisation, Multiobjective Genetic Algorithms, NSGA-II

1. INTRODUCTION

Project managers face multiple and conflicting decisions during the execution of a project in order to successfully develop it within the specified time span, budget and quality.

Among the decisions that need to be made in a software development project, estimating not only the average team size but the initial team size needed to develop the project can be placed among the most influential decisions regarding the productivity of the team and, eventually, the cost and time required to carry out the project.

Project team size has drawn a lot of attention over the years. Basically, large teams have been considered ineffective while small teams are perceived as better at delivering results. Brooks [7] already claimed in 1975 that assigning more programmers to a project running behind schedule will make it even later, due to the time required for the new programmers to learn about the project, as well as the increased communication overhead. Furthermore, team size does not remain stable throughout the project lifecycle. On the contrary, project planning needs to determine the initial team size and the policies and schedule required to add or remove

people to or from the initial team. Accordingly to Brooks, the loss of productivity suffered when staff are added directly affects key project performance indicators such as schedule, quality and cost.

Therefore, project managers need reliable ways to decide about the effects of their decisions regarding the rate of change in development teams on the software products, projects and processes in general. From the pioneering application of Forrester's System Dynamics (SD) simulation approach to software project modeling by Abdel-Hamid and Madnick [1], SD simulation modeling has been applied to many aspects of software development and management. Simulation enables project managers to build and run the models to better understand the implications of candidate project strategies and decisions.

However, a simple evaluation of the simulation outputs of a model is often not enough to determine the best decisions that maximise project performance. Usually, a more exploratory and in-depth study is required to determine the most suitable combination of decisions that lead the best project results. Simulation optimisation can be defined as the process of finding the best values of some decision variables for a system where the performance is evaluated based on the output of a simulation model of this system [20]. Currently, simulation optimisation functionalities are often found as part of simulation packages, being the metaheuristics approach among the most used methods for simulation optimisation. Metaheuristic techniques are a family of approximate (stochastic) optimisation algorithms that search iteratively in the solution space for a *good enough* solution.

However, while the approaches already implemented in the simulation packages often provide robust results when focussing on finding the optimal solution for an given objective, they do not usually provide the functionality of multiobjective optimisation, that is, simultaneously optimising two or more conflicting objectives. For instance, Vensim©[1], claimed to be the most used simulation tool for software project simulation modelling [25] uses the Powell hill climbing algorithm to search through the parameter space looking for the largest cumulative payoff. The payoff function is a user-defined function that is maximised or minimised and groups together the simultaneous objectives of the model user. Anylogic™[2] is a new tool in the arena of software project simulation. It uses the built-in OptQuest® optimiser to search for the best solution, given the objective function, constraints, requirements, and parameters that can be varied. Once again, it is the user who needs to provide a single objective function for maximisation or minimisation. Since software project management is a field where optimising conflicting objectives is one of the most frequent tasks that project managers need to face, it would be interesting to provide them with this facility.

This paper describes an approach that consists of using a multiobjective optimisation technique based on genetic algorithms for simulation optimisation in order to help software project managers to find the best values for initial team size and time estimates for a given project so that cost, time and productivity are optimised.

The phases carried out in this work are as follows. First, we developed a SD simulation model based on the literature and previous work. Second, we generated a database with all possible inputs combinations of the simulation runs. Although, in theory, the multiobjective genetic algorithm should call the simulation tool (i.e., the model corresponds to the fitness function) as many times as necessary while converging to the optimum values, this is not possible due to licence issues. Therefore, in the process of generating the database with input and output results, we have probably executed the simulation tool many more times than really necessary. Third, we ran several executions of the genetic algorithm with different multi-objectives as well as constraints obtained from the sensitivity analysis performed in the SD model.

The rest of this paper is structured as follows. Section 2 covers the background, Section 3 describes the SD model built to simulate a software project. Next, we describe the application of a multiobjective optimisation algorithm based on genetic algorithms with the simulation model in Section 4. Finally, Section 5 provides some conclusions and future research directions.

2. BACKGROUND

Abdel-Hamid and Madnick [1, 2] developed a highly aggregated simulation model of software project dynamics. The advantage of using System Dynamics is that one can experiment with different management policies before, during and after the execution of a project (post-mortem analysis) without additional cost. Among other things, their model was used to analyse Brooks' law applying different staffing policies on cost and schedule in a specific project, the NASA DE-A project. The authors conclude that adding more people to a late project always causes it to become more costly but does not always cause it to complete later. In this case, Brooks' law holds when the time to complete is less than 30 days (which would correspond to a project of approximately 24KDSI, 2,220 Person-days and 380 days).

The application of metaheuristic techniques to Software Engineering problems has generated a research field known as Search Based Software Engineering (SBSE) [13, 12]. So far, SBSE has been majoritarily applied to software testing problems [18] but it is been increasingly applied to other software engineering problems such as project management. For example, in Software Project Staffing, Di Penta *et al.* [9, 4] analysed Brooks' law using genetic algorithms. Alba and Chicano [3] have applied genetic algorithms as a technique to optimise people allocation to software development tasks. Zhang *et al.* [26] and Saliu and Ruhe [21] have applied metaheristic techniques in the next release problem, etc.

However, although simulation optimisation has been a productive topic of research in other fields, not many applications can be found in the field of software project management. Hanne and Nickel [11] considered the problem of planning inspections and other tasks within a software development project with respect to the objectives of quality, project duration, and costs. They built a discrete-event simulation model comprising the phases of coding, inspection, test, and rework and formalised the problem of project scheduling as a multiobjective simulation optimisation problem. Di Penta *et al.* [5] shows how search-based optimisation techniques can be combined with a queuing simulation model to address the problems of allocating resources to a software project and assigning tasks to teams.

As in the example above, most of the applications of sim-

[1] `http://www.vensim.com/`
[2] `http://www.xjtek.com/`

ulation optimisation that can be found in the field of software project management use the discrete-event paradigm as the simulation approach. However, there are also some applications of simulation optimisation using the System Dynamics simulation approach. For instance, Ng [19] reported an approach for integrating simulation and optimisation of System Dynamics models using Matlab® and Simulink® and demonstrated how to combine genetic algorithms, fuzzy logic expert input and System Dynamics modelling for improving decision-making. They applied their approach in the classical market growth model. Kremmel *et al.* [15] developed a System Dynamics simulation model to analyse the dynamics of city problems and city development under three types of policy interventions. They used genetic algorithms for maximising the benefits of policy decision making.

There are also some studies for optimising agent-based simulation models. Better *et al.* [6] describes work on progress consisting of incorporating advanced data mining techniques to identify relevant system inputs and to analyse the way these inputs interact within the system. The approach is applied in an agent-based simulation model for market research that works at both the consumer and the company level.

3. SIMULATION MODEL FOR SOFTWARE PROJECT MANAGEMENT

The simulation model has been built followings Law's methodology [16]. This section describes the model according to Kellner's proposal for simulation model description [14].

3.1 Model Purpose and Scope

The purpose of a simulation model can be described as the key questions the model has to address. In our case, the purpose of the simulation model is to help analyse the effect of uncertainty of both the schedule estimate and the initial team size of a software project on the key indicators of project success, namely time, cost and productivity. Determining the model scope is also an important issue, since the scope needs to be large enough to fully address the key questions posed. For the purpose of this work, the scope of the model is a software development project with a medium time-span and one project team.

We next sumarise the most important input and output variables.

3.2 Output Variables

The output variables are the information elements needed to answer the key questions that were specified along with the purpose of the model. For the purpose of this study, we will need the following outputs of the model:

- *Project End* ($Time$): The final time of the project.
- *Cumulative Cost* ($Cost$): The final cost of the project.
- *Productivity* ($Prod$): The average productivity reached by the team through the project lifecycle. This is calculated as the ratio between size (*Function Points -FP-* in this case) and the *Project End* (time taken to finish the project).

Other output variables that are helpful for analysis during the simulation timeframe are the following:

- *Fraction Complete*: The percentage of project completion at any time of the simulation.
- *Effective Workforce*: The effective work rate performed by the team.

3.3 Input Parameters

The input parameters to include in the model depend on the result variables desired and the process abstraction identified. To simulate a software project different input parameters are required, each of them customising the simulation model to both the features of the project and the organisation.

In our case, the model built provides input parameters to describe the features of the project under development such as the initial estimations of size and time, the quality level desired, the initial team size and its composition, the maximum workforce allowed, the wage rate, etc. In addition, the model also provides a set of input parameters to customise the model to the features of the organisation developing the project. Among these features, the following ones can be highlighted: hiring and dismissals delays, average time to overwork, the effect of fatigue on product quality and team productivity, etc.

For the sake of clarity, we will only describe here the input parameters that allow us to model decision making regarding the purpose of this study, that is, the initial team size and its composition, together with the initial estimations of project size and time to develop.

- *Initial Novice Workforce* ($NoviceWf$): The initial number of novice personnel allocated to the project.
- *Initial Experienced Workforce* ($ExpWf$): The initial number of experienced personnel allocated to the project.
- *Project Size* ($Size$): The estimate of project size (we consider Function Points -FP-[17] as a measure of the size).
- *Scheduled Time* ($SchldTime$): The estimate of project schedule.

3.4 Process Abstraction

When developing a simulation model, it is necessary to identify the key elements of the process, their interrelationships and behaviour, for inclusion in the model. The focus needs to be on those aspects of the process that are especially relevant to the purpose of the model and are believed to affect the result variables. The model developed is structured in three main subsystems:

- *Development*: This subsystem models the software development process excluding requirements, operation and maintenance.
- *Team management*: This deals with hiring, training, assimilation and transfer of the human resources. It includes Brooks' law to model training and communication overhead due to team size.
- *Control and Planning*: This subsystem provides the initial project estimates and models how and under what circumstances they will be revised through the software project life cycle.

Table 1: Control Parameters for Sensitivity Experiment

Input parameter	*Range*	*Step*
Initial Novice Workforce	[0-10]	1
Initial Experienced Workforce	[2-10]	1
Scheduled Completion Time	[45-80]	5

Under the System Dynamics simulation approach, all systems, no matter how complex, consist of networks of feedback loops, and all dynamics arise from the interaction of these loops with one another. Therefore, much of the work when building a System Dynamics model is discovering and representing the feedback loops, which along with stock and flow structures, time delays, and nonlinearities, determine the dynamics of a system. For the purpose of this study, the simulation model built consists of a network of 77 interacting feedback loops and 89 equations.

3.5 Sensitivity Experiment

Using the model described, we design a scenario for simulation and analysis of the sensitivity of the main output variables to the variation of the main input parameters.

Let us assume that the project size has been estimated as $500FP$ and that from the organisation historical data, the time required to develop a FP is 2 days. Therefore, the time scheduled for this project should be approximately 50 months. Let us also assume that in this particular project there are some new aspects that lead to the project manager to some uncertainty regarding the time estimation and the number of personnel that should be allocated for the initial team for this project.

In this context, a sensitivity experiment carried out with the simulation model can help the project manager to visualise the effect that under- or overestimating the project schedule, as well as the initial team size and composition between novice and experience personnel can have over the project final outcome.

Table 1 collects the values of the control parameters for the sensitivity experiment. In the experiment, the simulation model is run to obtain a database with all the output corresponding to each possible combination of the input parameters that control the experiment. Considering the sensitivity analysis, we are assuming that the minimum size of the development team is two experienced people. The number of experienced people can rise from 2 up to 10. Regarding the initial number of new personnel in the project, the values vary from 0 to 10. These restrictions will lead to designing a development team whose initial size is no larger than 20 people. As for the time estimates, the experiment allows for a range starting from 45 and up to 80 months.

Figure 1 shows the sensitivity of the output variable *FractionComplete*. When this output variable reaches 1 it means that the project is already finished since 100% of the tasks pending has been developed. According to this experiment, the final schedule of the project is within the range from 44 to 81 months.

Figure 2 shows the sensitivity of the output variable *EffectiveWorkforce*. This output variable represents the effective work rate the development team is able to achieve at every moment of each simulated project. It results from calculating the real productivity of each particular team taking into account their training and communication overheads.

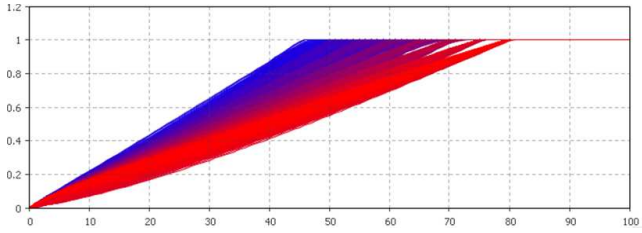

Figure 1: Sensitivity of the output variable *FractionComplete*

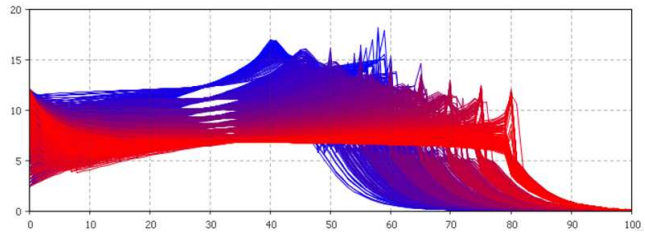

Figure 2: Sensitivity of the output variable *EffectiveWorkforce*

Figure 3 shows the sensitivity of the output variable *CumulativeCost*. This output variable collects the time evolution of the cost of each simulated project. As expected, the larger the team, the bigger the costs incurred in the project. The values of this output variable vary in a range from \$992K to \$2,551K.

Among the many managerial decisions that need to be made in a software project, personnel related factors are the ones affecting the productivity most [23]. This raises the concern about empirical evidence about the relationships between project attributes, productivity and staffing levels that can help optimise managerial decisions. Concretely, regarding the team size it is commonly acknowledged that the time spent in communication among team members increases with the size of the team. Project team size therefore affects schedule decisions, which are also acknowledged as an important factor in project success [24]. Furthermore, team size is important when making decisions about the structure of teams and the eventual partitioning of projects into smaller sub-projects. If an optimal team size could be found, then the decomposition of projects into smaller pieces becomes a key management practice with direct implications in the decision of distributing project teams.

3.6 Simulation Optimisation

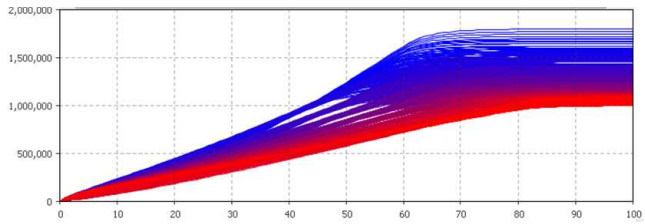

Figure 3: Sensitivity of the output variable *CumulativeCost*

Table 2: Input Parameter Values for Single Objective Optimisation

Output		Input Parameters		
		$NoviceWf$	$ExpWf$	$SchldTime$
$Cost$	\$992K	0	10	80
$SchldTime$	44.75	3	10	40
$Prod$	11.47	3	10	40

Once the sensitivity of the output variables of the model has been determined, the next step for the project manager should be to use the model in order to decide what values of the input parameters optimise the key project indicators. Current simulation tools provide their users with simulation optimisation but only for a single fitness function. That is, all the objectives need to be aggregated together to form a single objective or a scalar fitness function which is then treated by some classical techniques, mostly simulated annealing and scatter search.

This approach brings problems regarding how to normalise, prioritise and weight the different objectives in the global fitness function. In software project management it is also usual that conflicting objectives interact with each other in nonlinear ways. Therefore, finding an adequate function becomes the critical point in this approach since the set of solutions produced is highly dependent upon the function selected and the weights assigned.

In this section, we use the optimisation module built in Anylogic$^{\text{TM}}$ to optimise simulation output. We next show and discuss the results obtained in single and multiobjective simulation optimisation.

1. Single objective optimisation.

   In single objective optimisation, the tool finds the values of the input parameters that maximise or minimise a single output variable.

   Table 2 shows the values of the input parameters that optimise each single output variable according to the different optimisation experiments carried out using the simulation framework.

   It can be seen that the initial team size and the scheduled completion time vary depending on the objective one wants to achieve. Typically, this is not a very realistic situation in software project management, since project managers would be interested in the combination of input parameters that lead to the project with the maximum productivity and the minimum cost and time. Therefore, a multiobjective optimisation is needed.

2. Multiobjective simulation optimisation.

   When using current simulation frameworks for simulation optimisation such as Anylogic$^{\text{TM}}$, it is necessary to aggregate all the objectives into a single fitness function. This fitness function is then maximised or minimised, depending on the user request, mainly using scatter search.

   The simplest way to do this is to bundle all the objectives into a single fitness function using a linear function. For the purpose of this study, it was assumed that the project cost is the main objective driver and so different weights were used to determine how the two other objectives were related to the driving objective (Eq. 1).

$$\begin{aligned} Fitness(i) &= CummCost(i) \\ &+ \frac{1}{weightTime(i)} \cdot TimeProjEnds(i) \\ &+ \frac{1}{weightProd(i)} \cdot Prod(i) \end{aligned}$$

   The optimisation experiments carried out with the tool determined that the values of input parameters that optimise this fitness function are an initial development team of 10 experienced personnel ($ExpWf$) and 0 novice personnel ($NoviceWf$), and an initial schedule estimate of 80 months ($SchldTime$).

   The optimisation module of Anylogic$^{\text{TM}}$ concludes that no matter the weights, for a linear fitness function a development team formed by ten experienced people and a time estimate of 80 months is the best configuration possible to maximise productivity and minimising cost and development time. However, we have just seen through single objective optimisation that this combination of input parameters minimises cost, but does not minimise time nor maximises productivity.

   In the following section, we will apply and discuss the application of multi-objective optimisation techniques.

4. APPLYING NSGA-II TO THE SIMULATION DATA

As stated previouly, there are a large number of problems within the software engineering discipline that can be solved with metaheuristic techniques. In turn, there are multiple metaheuristic techniques available, and Multi-objective Optimisation problems (MOP) are those that involve multiple and conflicting objective functions. MOP is also known as Multiple Criterion Decision Making (MCDM) in other fields such as in operation research. In general, the solutions for MOP are defined using the Pareto front, which can be formally defined as follows.

Given the minimisation of n components $f_k, k = 1, \ldots, n$, of a vector function $\mathbf{f}$ of a vector variable $\mathbf{x}$ in $\mathcal{D}$, i.e., $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$ and subject to inequality and equality constraints ($g_j(\mathbf{x}) \geq 0, j = 1, \ldots, J$ and $h_k(\mathbf{x}) = 0, k = 1, \ldots, K$). A vector $\mathbf{u} = \{u_1, \ldots, u_k\}$ dominates a vector $\mathbf{v} = \{v_1, \ldots, v_k\}$, denoted by $\mathbf{u} \preceq \mathbf{v}$ if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$ (assuming the objective is always to minimise).

The set of non-dominated decision vectors, also known as *Pareto-optimal*, constitute the *Pareto front*, i.e., a set of solutions for which no objective can be improved without worsening at least one of the other objectives.

In this work, as multiobjective algorithm, we applied the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) developed by Deb *et al.* [8] as an extension of an earlier proposal by Srinivas and Deb [22]. The NSGA-II is a computationally efficient algorithm even with a large number of objectives and population size.

The population individuals are evaluated (assigned fitness values) in relation to how close they are to the Pareto front and a crowding measure. The fitness value according to its

non-domination rank is calculated as follows. The Pareto front is *Rank 1*. If we calculate a new Pareto front removing individuals in *Rank 1*, individuals in the new Pareto front form *Rank 2*, etc. Thus, individuals in lower ranks are given higher fitness values (as we are minimising). The NSGA-II algorithm also considers the sparsity (density) of the individuals belonging to the same rank using a crowding measure (the Manhattan distance among individuals), with the idea of promoting diversity within the ranks (the larger the sparcity, the better). In addition, the NSGA-II includes elitism in order to maintain the best solutions from the *Pareto front* found.

The NSGA-II algorithm is described in Algorithm 1, where the *sort()* function returns the individuals following a partial order, $(\prec_n)$, defined as: $i \prec_n j$ if $(i_{rank} < j_{rank}) \vee ((i_{rank} = j_{rank}) \wedge (i_{distance} > j_{distance}))$.

Algorithm 1 NSGA-II Algorithm [8]

1: $P_0 \leftarrow$ **makeInitalRandomPopulation**() ▷ Initial Population of size N
2: $Q_0 \leftarrow$ **makeNewPopulation**(P_0)
3: $R_0 = \emptyset \leftarrow \wedge\ t \leftarrow 0$
4: **while** $t \leq max\_generations$ **do**
5: $R_t \leftarrow P_t \cup Q_t$ ▷ Combine parent and offspring populations
6: $\mathcal{F} \leftarrow$ **fastNonDominatedSort**(R_t)
7: $P_{t+1} \leftarrow \emptyset \wedge i \leftarrow 1$
8: **while** $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do** ▷ While population size is not full
9: **crowdingDistance**$(\mathcal{F}_i)$ ▷ Calculate crowding measure in $\mathcal{F}_i$
10: $P_{t+1} \leftarrow P_{t+1} \cup F_i$ ▷ Include the ith rank into the population
11: $i \leftarrow i + 1$
12: **end while**
13: **Sort**$(\mathcal{F}_i, \prec_n)$ ▷ Sort in descending order using $\prec_n$
14: $_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ ▷ Fill population untill size N
15: $Q_{t+1} \leftarrow$ **makeNewPopulation**(P_{t+1}))
16: $t \leftarrow t + 1$
17: **end while**
18: **return** $\mathcal{F}_1$ ▷ Return the best *Pareto rank*

Multi-objective genetic algorithms complement simulation models, as we may want to optimise multiple parameters at the same time without making assumptions about which objective takes priority. In this work, we have used JMetal[3] [10], a metaheuristic algorithm framework that implements many of the current state of the art multiobjective genetic algorithms, including NSGA-II.

The results of the simulation were obtained using Anylogic^{TM}. However, as the tools cannot be integrated due to license restrictions, we generated and stored the results in a database. Then the JMetal framework was used to find the Pareto fronts considering different executions with multiple objectives, a population of 50 individuals and 200 iterations. Obviously, the generation of the whole dataset with the results in advance is not practical nor possible when considering continuous attributes or a large number of them, and the integration of both tools is necessary to consider the approach scalable.

Considering the SD simulation model described previously and two objectives, optimising *time* and $effort$, it can be observed that there is a large difference in cost between finishing the project at week 44.75 or 45.58 (the simulation output -*cost*- for the former *time* is $1,289K and $1,128K for the later) as shown in Table 3 and graphically in Figure 4. This is due to the fact that there are more personnel involved and after this point, it can be observed that the cost decreases if we increase the duration of the project using the same initial values for personnel. From the software engineering point of view is also interesting to observe that such an *elbow* in the Figure shows that there is a limit to the amount of time we can shorten a project.

[3] `http://jmetal.sourceforge.net/`

Table 3: Pareto Front for Two Objectives, *Time* and *Cost*

NoviceWf	*ExpWf*	*SchldTime*	*Time*	*Cost* ($(K))
3	10	40	44.75	1,289
2	10	45	44.85	1,287
1	10	45	45.58	1,128
0	10	55	54.90	1,053
0	10	60	59.84	1,035
0	10	65	64.79	1,021
0	10	70	69.72	1,010
0	10	75	74.71	1,001
0	10	80	79.68	992

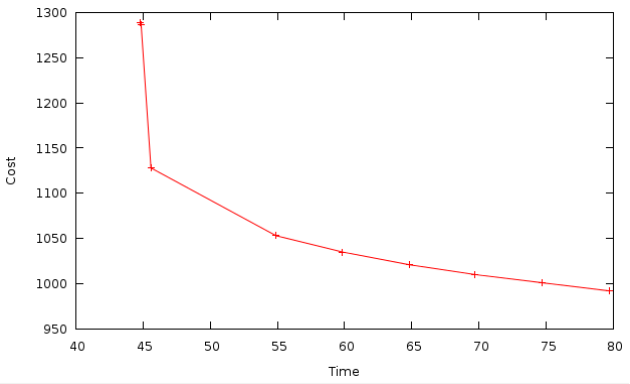

Figure 4: Pareto Front for Two Objectives

We obtained a bit more variety in the number of personnel with three objectives when we also consider the maximisation of productivity as an objective. The shape of the results is, however, very similar as it can be observed in Figure 5 showing the 3-dimensional representation of the Pareto front.

In both previous cases, the observed behaviour is that

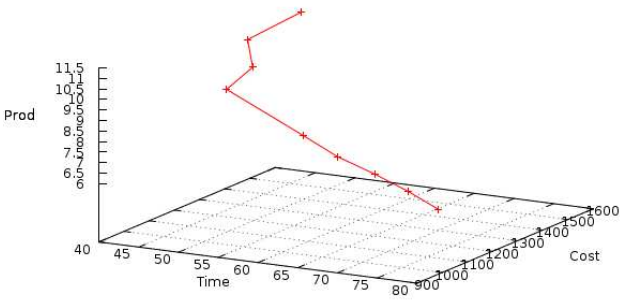

Figure 5: Pareto Front for Three Objectives, *Time*, *Cost* and *Prod*

Table 4: Pareto Front for Three Objectives, *Time*, *Cost* and *Prod*

NoviceWf	*ExpWf*	*SchldTime*	*Time*	*Cost* ($(K))	*Prod*
5	10	35	44.96	1,548	11.12
4	10	40	45.47	1,318	10.99
5	10	45	48.72	1,235	10.26
1	10	50	50.00	1,091	9.99
3	8	60	60.00	1,082	8.33
3	9	65	64.90	1,060	7.70
4	10	70	69.85	1,052	7.15
3	10	75	74.78	1,030	6.68
0	8	80	79.69	993	6.27
0	10	80	79.68	992	6.27

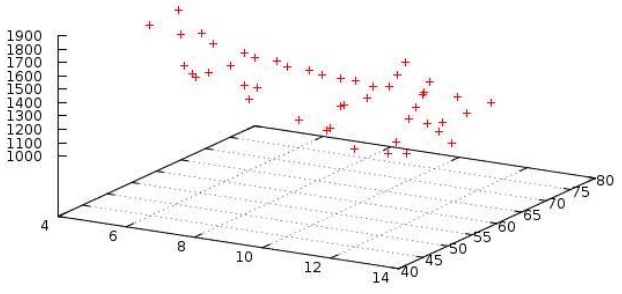

Figure 6: Pareto Front for Five Objectives

Pareto fronts obtained used the maximum number of experienced personnel allowed (*ExpWf*) or very close to it (this only changed when the differences in salary are around 5 fold). However, in the case of this not being possible or desirable (e.g., to distribute them among other projects, limited amount of experts, etc), we can add further objectives and/or constraints whilst searching for the Pareto front. For example, Figure 6 shows the Pareto front considering in addition to the previous objectives, the minimisation of both the number of experienced personnel and the addition of both novice and experienced personnel. The number of projects in the Pareto front increased. If constraints are used, for example, limiting the number of experienced personnel to certain value, the Pareto front obtained was always composed with projects using the higher value allowed.

When compared to single objective approaches, we can observe that with single objective solutions are close to the extreme in the range of solutions found in the Pareto front. The set of multiobjective solutions can help to analyse project trends and explain the trade-offs of applying different project policies to, for example, discover the amount of crunching we can perform in project as we have seen previously.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have applied a multiobjective optimisation technique to a System Dynamics model for project management. Multiobjective optimisation techniques applied to simulation models give project managers better control over the set of input variables than single optimisations. We have shown that using multobjective techniques can lead to achieve better results in terms of finding the input parameters that will maximise output parameters such as time, cost and productivity than single objective optimisations provided by current tools. This is due to the fact that there is no need to calculate the weights when combining the conflicting objectives into a single one. The range of solutions in the pareto front can also help with understanding different project policies.

As future work, we will apply multiobjective techniques to more complex models and compare different multiobjective approaches. Also, as the number of variables and solutions in the Pareto front increases with larger and more complex models (as well as with the number of objectives), we will explore visualisation and clustering techniques to present the results.

Acknowledgments

This research was partly supported by the Spanish Ministry of Science and Innovation and the European FEDER funds under projects TIN2007-67843-C06-04, TIN2010-20057-C03-03, TIN2011-68084-C02-00 and the Universities of Cádiz, Alcalá, Seville and Oxford Brookes University.

6. REFERENCES

[1] T. Abdel-Hamid and S. E. Madnick. *Software Project Dynamics: An Integrated Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

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

[3] E. Alba and J. F. Chicano. Software project management with gas. *Information Sciences*, 177(11):2380–2401, 2007.

[4] 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 Software Engineering*, 30(1):43–58, 2004.

[5] G. Antoniol, M. D. Penta, and M. Harman. The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study. *Software – Practice and Experience*, To appear.

[6] M. Better, F. Glover, and M. Laguna. Advances in analytics: Integrating dynamic data mining with simulation optimization. *IBM Journal of Research and Development*, 51(3.4):477 –487, May 2007.

[7] F. P. Brooks. *The Mythical Man-Month*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, anniversary ed. edition, 1995.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr. 2002.

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

[10] J. Durillo, A. Nebro, and E. Alba. The jMetal framework for multi-objective optimization: Design and architecture. In *IEEE Congress on Evolutionary Computation (CEC'2010)*, pages 4138–4325, Barcelona, Spain, July 2010.

[11] T. Hanne and S. Nickel. A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *European Journal of Operational Research*, 167(3):663–678, 2005. Multicriteria Scheduling.

[12] M. Harman. The current state and future of search based software engineering. In *Future of Software Engineering (FOSE'2007)*, pages 342–357, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[13] M. Harman and B. F. Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, 2001.

[14] M. I. Kellner, R. J. Madachy, and D. M. Raffo. Software rocess simulation modeling: Why? what? how? *Journal of Systems and Software*, 46(2-3):91–105, 1999.

[15] T. Kremmel, J. Kubalík, and S. Biffl. Software project portfolio optimization with advanced multiobjective evolutionary algorithms. *Applied Soft Computing*, 11:1416–1426, January 2011.

[16] A. M. Law. How to build valid and credible simulation models. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, 2008.

[17] C. J. Lokan. Function points. *Advances in Computers*, 65:297–347, 2005.

[18] P. McMinn. Search-based software test data generation: a survey: Research articles. *Software Testing, Verification and Reliability*, 14:105–156, June 2004.

[19] T. Ng, M. Khirudeen, T. Halim, and S. Chia. System dynamics simulation and optimization with fuzzy logic. In *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2009)*, pages 2114–2118, 2009.

[20] S. Ólafsson and J. Kim. Simulation optimization: simulation optimization. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, WSC '02, pages 79–84, 2002.

[21] M. O. Saliu and G. Ruhe. Bi-objective release planning for evolving software systems. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ESEC-FSE'07, pages 105–114, New York, NY, USA, 2007. ACM.

[22] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248, September 1994.

[23] A. Trendowicz and J. Münch. Factors influencing software development productivity – state-of-the-art and industrial experiences. Elsevier, 2009.

[24] J. Verner, W. Evanco, and N. Cerpa. State of the practice: An exploratory analysis of schedule estimation and software project success prediction. *Information and Software Technology*, 49(2):181–193, 2007.

[25] H. Zhang, B. Kitchenham, and D. Pfahl. Software process simulation modeling: An extended systematic review. In J. Münch, Y. Yang, and W. Schäfer, editors, *New Modeling Concepts for Today's Software Processes*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

[26] Y. Zhang, M. Harman, and S. A. Mansouri. The multi-objective next release problem. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, volume 1, pages 1129–1136, London, UK, July 2007. ACM Press.