Statistical Methods & Applications (2004) 13: 227–239
DOI: 10.1007/s10260-004-0080-8

SMA

Optimal experiments in the presence of a learning effect: a problem suggested by software production

Alessandra Giovagnoli[1] **and Daniele Romano**[2]

[1] Dipartimento di Scienze Statistiche, Università di Bologna, Via Belle Arti, 41 – 40126 Bologna, Italy (e-mail: giovagno@stat.unibo.it)
[2] Dipartimento di Ingegneria Meccanica, Università di Cagliari, Piazza d'Armi – 09123 Cagliari, Italy (e-mail: romano@dimeca.unica.it)

Abstract. In software engineering empirical comparisons of different ways of writing computer code are often made. This leads to the need for planned experimentation and has recently established a new area of application of DoE. This paper is motivated by an experiment on the production of multimedia services on the web, performed at the Telecom Research Centre in Turin, where two different ways of developing code, with or without a *framework*, were compared. As the experiment progresses, the programmer's performance improves as he/she undergoes a learning process; this must be taken into account as it may affect the outcome of the trial. In this paper we discuss statistical models and D-optimal plans for such experiments and indicate some heuristics which allow a much speedier search for the optimum. Solutions differ according to whether we assume that the learning process depends or not on the treatments.

Key words: D-optimal design, Learning effect, Software engineering

1 Planning experiments for software production

In *software engineering*, namely the software production process, different methods of developing code need to be compared on an empirical basis. This sector is atypical with respect to those that historically have seen the growth of Experimental Design but the need to plan experiments in this field may open up a new and very significant area of application of DoE. These applications are very recent – the first book ever on the experimental approach to software engineering is Wohlin et al. (2000) – and give rise to problems of a non-trivial nature for the applied statistician, both for theoretical and economic reasons. A broad discussion on using experiments in software engineering is given by Tichy (1998) and by Zelkowitz and Wallace (1998).

Technical peculiarities are the strong incidence of the human factor linked to the programmers and the length of time required for writing code, during which the conditions of the experiment may possibly change, e.g. because of greater familiarity of the programmer with the tools and with the context of the applications, interactions with the team, etc. The main economic constraints are the strong competitive pressure from the software industry and the rapid obsolescence of its products. The development planning time is so short that systematic work on prototypes under controlled conditions is not economically feasible even for medium-to-large sized enterprises. The consequence is that the experiments performed in an industrial environment are mainly small ones. Most empirical studies in software engineering fall into two categories: controlled experiments with students in academic settings, or case studies with professional programmers in an industrial setting. The former allow for controlling variables, but have two serious drawbacks: students do not behave like professional programmers in the workplace (because of inexperience, pressure, interruptions, conflicts, hierarchies) and projects performed by students cannot but be limited in size and scope; thus industrial users are often sceptical in extrapolating the results to an industrial context. The other studies involve few programmers, or even just one, with the task of developing different applications, as industrial users rarely commit resources to replicate development projects which would guarantee reliability of experimental results. For a critique of empirical studies with students and a discussion of single programmer experiments see Harrison (2000).

Our motivating example is an experiment on the production of multimedia services on the web, carried out in 1999 at the Telecom Research Centre in Turin (Morisio et al., 2002), over a period of 6 months with interruptions, with the purpose of evaluating two different ways of developing code. We describe the actual experiment first, and then look for D-optimal designs for experiments of the same type. The solutions differ according to whether we assume that the programmer's learning process depends or not on the development method employed.

2 An experiment on software development for the production of services on the internet

The object of the investigation of Morisio et al. (2002) was the production of software based on a previously developed software infrastructure called *framework*. This is a promising alternative to the traditional development from scratch. Application frameworks are a reuse technique meant to exploit commonality in the engineering of product lines, namely software production over several releases for different customers. Frameworks are semi-complete applications: the framework is developed first, then several applications are derived from it. The research question is whether adoption of framework-based development is convenient in terms of programmer productivity and software quality. All applications supported the delivery of a service on the Internet. Services in the experimental group provided on-line environmental monitoring by a spy camera, a multiconference session, a distance learning session, an on-line auction and videos on demand. Applications

in the control group were a variant of a previously developed application in the experimental group. The experimental responses were chosen to be:

- productivity Y_P, namely net size divided by the effort required for its completion, measured by *Object Oriented Function Points per hour*, (OOFP, see Caldiera et al., 1998);
- an index $Y_{(Q)}$ of quality of programming, calculated as the percentage ratio between rework effort required to correct code failures identified by the acceptance test and development effort (0% means no failure encountered, 100% means that correction took as long as development).

The objective of the experiment was to test the following hypotheses:

1 *Development with framework provides higher productivity than development without framework*. The hypothesis was whether in the same time interval the programmer is able to write a larger piece of code if he/she uses the framework. The size considered in the experiment is the net size, or size of the actually written code. The framework size, practically reused *in toto* in the experimental group, obviously does not come into consideration.
2 *Learning has a greater impact on productivity in development with framework than in development without framework.*
3 *Development with framework is less prone to failures than traditional development.*
4 *Learning has a greater impact on failure reduction in development with framework than in development without framework.*

Another objective of the study was the building of predictive models for both responses. In practice these models are mandatory for forecasting the break-even point between the two development modes (in terms of number of developed services or cumulated development time). In fact, this is a key item of information for a sound managerial decision about the production technology to adopt.

The industrial end user of this study wanted to perform an experiment in his environment as he did not trust experiments with students. On the other hand the company did not want to incur additional costs for replication and provided a single programmer only (a computer science graduate with good knowledge of the application domain and expertise in object oriented programming, Java beginner). *The programmer was not involved in developing the framework.*

In the experiment, using or not using the framework was the only treatment (controlled independent variable). Since the same subject developed all the applications, his performance was prone to improving as the experiment progressed because of some form of learning (the language, the application context, the treatments). To take this into account, a simple regression model was formulated for both responses, productivity $Y_{(P)}$ and quality $Y_{(Q)}$, which included a variable (X) indicating the cumulative size of the software developed up to each experimental run; thus X_t was the sum of the lengths $x_s (s = 1, 2, ..., t)$ of code relative to each application developed up to time t.

In performing the experiment the programmer developed five applications starting from the framework and four variants without it. The run order was randomized

but there were severe constraints that led to a highly restricted form of randomization, namely since the applications developed without the framework were variants of the ones with framework, the former were constrained to be developed after the ones with framework. Furthermore, two applications of similar size and complexity were both developed with the framework, one at the beginning and one at the end of the experiment. The outcome of the experiment, which is discussed in Morisio et al. (2002), suggested that productivity was best described by a model with an interaction between the learning effect and the use of a framework, while quality was best described by a model without interaction. Another covariate, code complexity, was introduced, measured by the number of implemented methods per size unit: it was found to be not statistically significant in describing quality but significant for productivity.

3 The optimal experimental design problem

3.1 The problem

We wish to formalise the type of experiment highlighted above with a view to choosing a good experimental plan. Let productivity and quality be measured as explained in Sect. 2. In the case-study experiment of Morisio et al. (2002) the response at time t was assumed to be dependent on the total quantity of code developed up to that moment. We follow that approach and express this dependence by means of a linear regression model in which the value of the regressor at time t is the *cumulated* value of the code lengths x's of all the applications developed up to time t. Different applications have different code length x and the values of the x's may or may not be known *a priori* (although it is usually possible to make reasonable assumptions on their order of magnitude and their ranks), thus the problem has various degrees of difficulty. We shall assume that the $x's$ are known prior to the experiment.

Assume there are v different treatments $\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_v$ and n applications (one for each observation) which play the role of experimental units. Let τ_i $(i = 1, \ldots, v)$ be the effects of the treatments. The experiment will consist in choosing a run order in which to develop the applications and matching one of the treatments to each application. Let x_t and $i(t)$, respectively be the code length of the application used and the treatment chosen at run t, and let X_t be the cumulative code length, $X_t = \Sigma_j x_j$ $(j = 1, ..., t)$. We stress that different orderings of the runs give rise to different values of the regressor at the $t-$th observation.

3.2 The model

For the purpose of the design we can assume that the response is described by either of the following simple linear regression models – according to whether we assume absence or presence of interaction between the treatment and the possible learning

effects:

$$\text{Model I}: \quad Y_t = \tau_{i(t)} + aX_t + \varepsilon_t \quad (t = 1, ..., n)$$
$$\text{Model II}: Y_t = \tau_{i(t)} + a_{i(t)}X_t + \varepsilon_t \ (t = 1, ..., n)$$

with ε_t normal independent errors with 0 mean and constant variance. Both models tend to oversimplify the description of the process, but this is unavoidable in the first ever investigation of the problem. We also wish to underline that

i) x does not have the status of a random variable, so that the cumulative values, the X's, are not random either.
ii) Dependence on the past is only assumed within the deterministic component of the model, not the stochastic one. A very similar situation arises in the designs of "cross-over trials" (see for instances Jones and Kenward, 1989). Since we assume that the ε_t's are "pure" random errors the responses in our model are not correlated (in a stochastic sense).
iii) A different model with correlated observations could reasonably arise should we experiment with more than just one programmer and regard the programmer's effect as random. In that case the model would be a variance component one, and we would assume constant positive correlation among responses involving the same programmer.
iv) Some covariates only become known during the experiment; this is what happened in the case study with the code complexity. We can make use of them for a possibly better analysis of the data, but since they are not known at the design stage, they cannot contribute to the construction of the design. This is the reason for not including such covariates in models I and II.
v) Our problem is non-standard, in that the values of the regressor X depend on the order in which the runs are performed. Commercial statistical packages, e.g., SAS, do not include such a possibility.

3.3 The design

We can design the experiment as follows:

Step 1:	list the statistical units in *increasing* order of their x-value
Step 2:	apply a one-to-one map ω of the units (i.e. the ordered x's) onto the set of integers T= {1,2, . . . ,n} (the run orders)
Step 3:	map the units onto the set of treatments.

Observe that Step 3 may or may not be under the experimenter's control. In the case study described above, the two treatments were presence or absence of the *framework* and the applications developed without *framework* had much smaller codes than those developed with *framework*. In such experiments, however, with a pre-determined link between the treatments and the x-values, there is an objective danger of *confounding* the treatment effects with the unit effects and therefore they should be avoided.

As a result of Steps 2 and 3, we find that each run $t \in T$ is uniquely mapped to a value of the regressor X on one hand and to a treatment on the other. Inverting this function, we find that to each treatment there correspond certain runs, and the associated values of the regressor. We can write:

$X_{ij} =$ regressor relative to the j-th observation in which the i-th treatment is used.
$\bar{X}_{i\bullet} =$ average of all the regressor values relative to the observations when the i-th treatment is used.
$\bar{X}_{\bullet} =$ average of all the values of the regressor.

3.4 The optimality criterion

Assume we are interested in estimating parameters, as well as testing hypotheses, under either Model I or Model II. Then we would choose an experimental design as near as possible to a D-optimal one, since under normality conditions and independent observations this criterion maximises the power of the tests and minimises the area of the confidence ellipsoids for the estimates (see for instance Atkinson and Donev, 1992).

Denoting by A_I the design matrix for Model I and A_{II} that for Model II, by the independence assumption of the observations the respective information matrices are $A_I^T A_I$ and $A_{II}^T A_{II}$ and their determinants can be written in terms of the regressor X as

$$\det\left(A_I^T A_I\right) = \left(\prod_{i=1}^{v} n_i\right) \sum_{i=1}^{v} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_{i\bullet})^2 \tag{1}$$

$$\det\left(A_{II}^T A_{II}\right) = \left(\prod_{i=1}^{v} n_i\right) \prod_{i=1}^{v} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_{i\bullet})^2 \tag{2}$$

(for the proof see Appendix 1). Note that (1) is the product of two terms: one is the product of the number of treatment replications $\left(\prod_{i=1}^{v} n_i\right)$, the other is the sum of the deviances of the values X_{ij} relative to each treatment. Similarly (2) is the product of the numbers of treatment replications by the *product* of the deviances of the regressor X relative to each treatment. Expressions (1) and (2) show that for a given total number n of observations, in order to maximise $\left(\prod_{i=1}^{v} n_i\right)$ we should use an equal or as equal as possible number of replicates of the treatments; equireplicating treatments is common practice and in this case it is also, everything else being equal as regards the regressor, a requirement for optimality. Thus from now on we shall make an additional assumption, namely that all our designs are (nearly)-equireplicated, i.e., $\exists h (2 \leq h \leq v)$ such that

$$n_1 = n_2 = \ldots = n_h = n_{h+1} + 1 = \ldots = n_v + 1.$$

(Clearly fully equireplicated designs are a special case when $h = v$).

Given the values of x, a brute force solution to the problem of finding the D-optimal design, at least in the class of all the (nearly)-equireplicated ones, would be to list all possible increasing sequences of X_t for all possible equireplicated treatment assignments and choose the sequence with the largest determinant of the information matrix. The second author has written a programme which does exactly that; however, even for not-so-complex experiments, e.g., $v = 2$ and $n = 9$, the computing time is impracticable on a medium size computer. It is therefore useful to be able to find an analytical solution at least in some special cases; this may, in the general case too, suggest a heuristic strategy for the search of the optimum more efficient than the simply enumerative one.

4 Some solutions for Model I

4.1 Sufficient conditions for optimality

Because of the equireplication assumption, maximisation of (1) reduces to maximizing

$$\mathrm{SS_W} = \sum_{i=1}^{v} \sum_{j=1}^{n_i} \left(X_{ij} - \bar{X}_{i\bullet}\right)^2 \tag{3}$$

which can be regarded as the within-treatments sum of squares of the regressor. In other words, the values of the regressor that correspond to the same treatment should be as diversified as possible. Because of the identity

$$\begin{aligned} \mathrm{SS_W} &= \sum_{i=1}^{v} \sum_{j=1}^{n_i} \left(X_{ij} - \bar{X}_{i\bullet}\right)^2 \\ &= \sum_{i=1}^{v} \sum_{j=1}^{n_i} \left(X_{ij} - \bar{X}_{\bullet}\right)^2 - \sum_{i=1}^{v} n_i \left(\bar{X}_{i\bullet} - \bar{X}_{\bullet}\right)^2 \\ &= \mathrm{SS_{TOT}} - \mathrm{SS_B}, \end{aligned} \tag{4}$$

the conditions:

$$\mathrm{SS_{TOT}} \text{ is a maximum} \tag{5}$$

and

$$\mathrm{SS_B} = 0, \text{ i.e. } \bar{X}_{1\bullet} = \bar{X}_{2\bullet} = \ldots = \bar{X}_{v\bullet} \tag{6}$$

are jointly sufficient for the design to be D-optimal for Model I. Condition (6), taking the averages of all regressor values relative to the same treatment to be all equal, is known as *balancing the covariates* (see for instance Giovagnoli, 1988). We observe that

---

$\mathrm{SS_{TOT}} = \sum_{i=1}^{v} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_{\bullet})^2 = \sum_{t=1}^{n} (X_t - \bar{X}_{\bullet})^2$ does not depend on the treatment allocations but only on the run order.

Thus we could choose to maximise $\mathrm{SS}_{\mathrm{TOT}}$ first and then look for a treatment assignment that makes $\mathrm{SS}_{\mathrm{B}} = 0$. Maximisation of $\mathrm{SS}_{\mathrm{TOT}}$, the deviance of regressors, can be done as follows: 1) maximise the range $R_X = X_n - X_1$; 2) pull the regressors as far as possible towards the extremes of the range, in such a way that the average is roughly the mid-point. Let $x_{(h)},\ h = 1, ..., n$, denote the non-decreasing sequence of covariates. Since the run order of the covariates x affects all the regressors but the last one, $X_n = \sum_{k=1}^{n} x_k$, in order to have the largest range we must minimise X_1. So we choose $X_1 = x_{(1)}$. Then objectives 1) and 2) are simultaneously met by assigning the remaining covariates in the run order

$$x_{(2)}, x_{(4)}, ..., x_{(n-2)}, x_{(n)}, x_{(n-1)}, ..., x_{(5)}, x_{(3)}\,,\ \ n \text{ even;}$$

$$x_{(2)}, x_{(4)}, ..., x_{(n-1)}, x_{(n)}, ..., x_{(5)}, x_{(3)}\,,\ \ n \text{ odd;}$$

or, equivalently, in the reverse order.

4.2 The special case when all x's have the same value

A very special case is when all the x's are equal to a common value x_0. Then the k-th observation in time has regressor $X_k = kx_0$ irrespective of the run order. Hence $\mathrm{SS}_{\mathrm{TOT}}$, the total deviance of the regressor, is a constant.

$$\begin{aligned}\mathrm{SS}_{\mathrm{TOT}} = \sum_{i=1}^{v}\sum_{j=1}^{n_i}\left(X_{ij} - \bar{X}_{\bullet}\right)^2 &= \sum_{i=1}^{v}\sum_{j=1}^{n_i} X_{ij}^2 - n\bar{X}_{\bullet}^2 \\ &= \sum_{k=1}^{n} k^2 x_0^2 - \frac{n(n+1)^2}{4} x_0^2.\end{aligned}$$

Thus condition (6) alone is sufficient for D-optimality. Approaching condition (6) as much as possible amounts to a "classical" numerical problem: that of partitioning the integers $1, 2, \ldots, n$ (i.e., the run orders) into v groups, one for each treatment, of (nearly) equal size with (nearly) equal averages. Table 3 shows some solutions, where A, B and C denote treatments (clearly the treatment labels should be randomized) and the sequence in each row denotes the order in which the treatments are used in the experiment.

4.3 Remarks on the general case

In the general case it is no longer true that $\mathrm{SS}_{\mathrm{TOT}}$ is a constant. However, we can apply the same strategy, namely maximize $\mathrm{SS}_{\mathrm{TOT}}$ first and then look for a treatment assignment that makes the design balanced, or approximately balanced. An example will illustrate this procedure:

Example. Assume two treatments, denoted by A and B, $n = 4$ and two of the applications with code length x_0 and the remaining ones with code $x_0 + d$, $d > 0$.

Table 3. Some D-optimal designs for Model I when the x's are all equal

$v = 2, n = 6$	A,B,A,B,B,A
	A,B,B,A,A,B
$v = 2, n = 8$	A,B,B,A,B,A,A,B
	A,B,B,A,A,B,B,A
	A,A,B,B,B,B,A,A
$v = 2, n = 9$	A,B,A,B,A,B,A,B,A
	A,B,B,A,A,A,B,B,A
	A,A,B,B,A,B,B,A,A
$v = 3, n = 9$	A,B,C,B,C,A,C,A,B
	A,B,C,C,A,B,B,C,A
$v = 3, n = 10$	A,B,C,B,C,B,A,C,A,B
	A,B,C,C,B,A,A,A,C,B
	A,B,C,A,C,B,A,B,C,A
$v = 3, n = 12$	A,B,C,B,C,C,A,A,B,A,B,C
	A,B,C,C,B,A,C,B,A,A,B,C
	A,B,A,B,C,C,C,C,B,A,B,A

It can be checked that $\mathrm{SS_{TOT}}$ is maximised by observing x_0 first and by taking $x_0 + d$ as the third observation. Thus both run sequences

$$x_0\ x_0 \quad x_0 + d \quad x_0 + d \quad \text{and} \quad x_0 \quad x_0 + d \quad x_0 + d \quad x_0$$

maximise $\mathrm{SS_{TOT}}$. We cannot choose a treatment allocation that satisfies (8) perfectly but near balance of the regressor, x_0, $2x_0$, $3x_0 + d$, $4x_0 + 2d$ and x_0, $2x_0 + d$, $3x_0 + 2d$, $4x_0 + 2d$ respectively, is achieved in both cases by the treatment sequence A,B,B,A.

In general, we underline that conditions (5) and (6) may not be simultaneously achievable, however a computer search may be greatly speeded-up by looking for the maximum $\mathrm{SS_W}$ within the set of run sequences with highest values of $\mathrm{SS_{TOT}}$. By implementing this idea, it is possible to find good designs for higher values of n ($n > 8$). As an example, the optimal design with x values as in the case study was found easily ($n = 9$): it is the sequence A,A,B,B,B,B,B,A,A. This calculation is not based on the observed x values but on their evaluation at a pre-experimental stage, namely the assumption that the applications developed without *framework* (treatment A) and those with it (treatment B) have $x-$values in a $1 : 10$ ratio.

5 Designs for Model II

5.1 Sufficient conditions for optimality

The problem is maximisation of

$$\prod_{i=1}^{v} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_{i\bullet})^2. \tag{7}$$

A set of conditions jointly sufficient for D-optimality is:

$$\mathrm{SS_W} \text{ is maximum}$$

and

$$\text{the deviances of the regressor relative to each treatment are all equal} \qquad (8)$$

Splitting the first condition into two further conditions as in 4.1 (conditions (5) and (6)):

$$\mathrm{SS_{TOT}} \text{ is maximum and } \mathrm{SS_B} = 0,$$

we obtain that conditions (5), (6) and (8) are jointly sufficient for optimality.

5.2 Implementing the sufficient conditions

As in 4.2, $\mathrm{SS_{TOT}}$ does not depend on the treatment allocation and the search for the optimal design can be divided into optimising run order first, and assigning the treatments afterwards.

In the special case when the x's are all equal $\mathrm{SS_{TOT}}$ is a constant, so that condition (5) does not apply. In this case, we can find optimal designs when $v = 2$ and n is even by the following method: it is sufficient to partition the integers $1, 2, \ldots, n$ (which stand for the run orders) into 2 equal size groups with (almost) equal sums. Assigning each treatment to all the runs in one group approximately ensures condition (6), as in 4.2, and if the assignment is such that central anti-symmetry of the treatment allocation in the run sequence holds, also condition (8) is satisfied. An example will help illustrate the idea.

Example. Assume only two treatments, denoted by A and B, $n = 6$ and assume all the applications have code length x_0. The values of the regressor are x_0, $2x_0$, $3x_0$, $4x_0$, $5x_0$, $6x_0$ and $\bar{X}_\bullet = 3.5x_0$. Since $1+4+5 \cong 2+3+6$, $1+3+6 \cong 2+4+5$ and $1+4+6 \cong 2+3+5$, all the allocations

(i) treatment A to runs 1, 4 and 5, treatment B to runs 2, 3 and 6
(ii) treatment A to runs 1, 3 and 6, treatment B to runs 2, 4 and 5
(iii) treatment A to runs 1, 4 and 6, treatment B to runs 2, 3 and 5

achieve near-balance. However, only the sequence (i) A,B,B,A,A,B, because of its central anti-symmetry, ensures equality of the within-treatment deviances of the regressor:

$$(x_0 - 3.33x_0)^2 + (4x_0 - 3.33x_0)^2 + (5x_0 - 3.33x_0)^2$$

and

$$(2x_0 - 3.67x_0)^2 + (3x_0 - 3.67x_0)^2 + (6x_0 - 3.67x_0)^2.$$

Table 4 shows a number of designs obtained in this way.

In all the other cases it may not be so simple to apply conditions (5), (6) and (8), but we can use them to drive an efficient computer search for good designs. This has been done by the second author. Appendix 2 contains an outline of his computer program.

Table 4. Some D-optimal designs for Model II when the x's are all equal and $v = 2$

$v = 2, n = 6$	A,B,B,A,A,B
$v = 2, n = 8$	A,B,B,A,B,A,A,B
$v = 2, n = 10$	A,B,B,A,B,A,B,A,A,B A,B,B,A,A,B,B,A,A,B
$v = 2, n = 14$	A,B,A,B,B,A,B,A,B,A,A,B,A,B A,B,B,A,A,B,B,A,A,B,B,A,A,B
$v = 2, n = 16$	A,B,B,A,A,B,B,A,B,A,A,B,B,A,A,B A,B,A,B,B,A,B,A,B,A,B,A,A,B,A,B A,A,B,B,B,B,A,A,B,B,A,A,A,A,B,B

6 Discussion

The design problem outlined in Sect. 3 seems to be new. This research is at a seminal stage and we have merely sketched out possible solutions in a number of simple cases. There are more complex features that call for future investigation to shed more light on the problem. We think the problem deserves to be better understood because the findings are potentially applicable in other contexts.

We end with some comments on the "goodness" of the experiment in Morisio et al. (2002). The design that was used in the experiment is not D-optimal either under Model I or Model II, since the actual treatment sequence was A, A, B, A, A, B, B, B, A while the one maximizing the determinant of the information matrix for both models is A, A, B, B, B, B, B, A, A. The sequence used in the experiment has D-efficiency $= 0.63$ under Model I and $= 0.75$ under II. The comparison between the two designs, however, is not fair, since we are comparing an experiment which was partly randomized with a non-randomized, hypothetical one.

We would like to conclude with a general consideration on the use of DoE as a strategic tool for empirical research in software production. Experiments on software development are a time-demanding activity and often require a huge investment in human labour: this is hardly compatible with the constraints imposed by the software market. Nevertheless, even in such circumstances, planned experiments have a significant role to play. A challenging solution would be for experiments to be fitted onto the ongoing production flow, each experiment being designed on the basis of knowledge acquired from the previous ones and therefore being able to guide the future setting of process parameters. This concept is inspired, in general, by the principles of sequential experiments.

Acknowledgements. Both authors are members of PRO-ENBIS, a project supported by funding under the European Commission's Fifth Framework 'Growth' Programme via the Thematic Network "Pro-ENBIS"; contract reference: G6RT-CT-2001-05059. Its mission is

- to promote the widespread use of sound science-driven applied statistical methods in European business and industry
- to facilitate the rapid transfer of statistical methods and related technologies to and from business and industry.

The authors are solely responsible for the content and it does not represent the opinion of the Community, the Community is not responsible for any use that might be made of data therein.

Appendix 1

Proof of (1) and (2)

First of all we point out that $\det(A_I^T A_I)$ is invariant under row permutations of the matrix A_I, thus we can order the rows of A_I according to the treatments, which amounts to writing

$$\text{Model I} : Y_{ij} = \tau_i + \alpha X_{ij} + \varepsilon_{ij} \ (i = 1, \ldots, v; j = 1, \ldots, n_i)$$

Now we reparametrize Model I as follows:

$$\text{Model } \tilde{\text{I}} : Y_{ij} = \tilde{\tau}_i + \alpha(X_{ij} - \bar{X}_{i\bullet}) + \varepsilon_{ij} \ (i = 1, \ldots, v; j = 1, \ldots, n_i)$$

where

$$\tilde{\tau}_i = \tau_i + \alpha \bar{X}_{i\bullet} \ \ (i = 1, \ldots, v). \tag{9}$$

Let $\tilde{A}_I$ be the design matrix of Model $\tilde{\text{I}}$. and denote by C the linear transformation of the parameters given by reparametrization (9). C is a $(v+1) \times (v+1)$ matrix such that $\tilde{A}_I C = A_I$ and $detC = 1$. Therefore $det(A_I^T A_I) = det(\tilde{A}_I^T \tilde{A}_I)$, so w.l.o.g. we assume Model $\tilde{\text{I}}$. The columns of $\tilde{A}_I$ are orthogonal and $\det\left(A_I^T A_I\right) = \left(\prod_{i=1}^{v} n_i\right) \sum_{i=1}^{v} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_{i\bullet})^2$ follows easily. The proof of (2) is similar.

Appendix 2

The heuristic optimisation algorithm to find good designs for the present problem is based on a local search method, called *tabu search* (see Glover et al., 1993). The general scheme is the following: an initial solution is found by implementing the sufficient conditions for D-optimality (5), (6) for Model I or (5), (6) and (8) for Model II, choosing as initial ordering for the treatments the base sequences

$$\begin{array}{ll} \mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_v, \mathrm{T}_v, \ldots, \mathrm{T}_2, \mathrm{T}_1 & \text{for Model I;} \\ \mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_v & \text{for Model II;} \end{array}$$

to be repeated until all units are assigned.

This solution is recorded as the first one in a "best solution" list (tabu list). Subsequent solutions are then generated from the current "best solution" by applying a set of previously defined perturbations, namely:

Exchange of adjacent treatments;
Circular shift of the treatments to the left;
Circular shift of the treatments to the right.

and picking the best of the perturbed solutions. The new "best solution" is added to the tabu list if not already in it, otherwise we choose the "second best" and so on. These steps are iterated and we stop 50 steps after no further improvement occurs.

References

Atkinson A C, Donev A N (1992) Optimum experimental design. Clarendon Press, Oxford

Caldiera G, Antoniol G, Fiutem R, Lokan C (1998) Definition and experimental evaluation for object oriented systems. In: Proceedings of Fifth IEEE Int. Symp. On Software Metrics, *Metrics '98*, Bethesda

Giovagnoli A (1988) Bilanciamento in Presenza di Covariate per un Modello Lineare. In: Proceedings of the XXXIV Scientific Meeting of the Italian Statistical Society, Siena, 27–30 April 1988

Glover F, Laguna M, Taillard E, De Werra D (eds) (1993) Tabu search. Baltzer, Amsterdam

Harrison W (2000) $N = 1$, an alternative for software engineering research? In: Workshop beg borrow or steal: using multidisciplinary approaches in empirical software engineering research. Int. Conf. on Software Engineering. Limerick, Ireland

Jones B, Kenward M G (1989) Design and analysis of cross-over trials. Chapman and Hall, London

Morisio M, Romano D, Stamelos I (2002) Quality, productivity and learning in framework-based development: an exploratory case study. IEEE Transaction of Software Engineering 8(9): 876–888

Tichy W F (1998) Should computer scientist experiment more? IEEE Computer, May: 32–40

Wohlin C, Runeson P, Höst M, Ohlsson M C, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer, Norwell, USA

Zelkowitz W F, Wallace D R (1998) Experimental models for validating technology. IEEE Computer, May: 23–31