Measurement and the software development process

Colin Kirsopp

Abstract

This paper argues that software measurement and the software development process are mutually dependent. Firstly, a high-level view of the relationship between measurement and the development process is taken. This view shows where the various activities within software measurement are positioned within the context of the high-level project stages. The topic of process modelling is introduced, and is used to develop a lower-level view of measurement integration with process. This shows that process models for measurement integration should be sufficiently fine-grained to show when measurements should be collected, how they should be collected, who should collect then, and how the results will be used (the usual lifecycle models are too coarse-grain to achieve this). It is also shown that measurement can be most easily integrated within an automated environment, as measurement collection should itself be automated.

The initial work on the integration of measurement with process shows that there are elements of measurement, such as measurement definition and empirical validation, that must be considered outside of an individual project. To facilitate these cross-project measurement activities, the organisational structure within which the individual projects run is examined. To illustrate this, the authors discuss the use of measurement within the organisational model provided by the TAME project [1-3].

1. Introduction

Software measurement is often considered in isolation from the software development process within which the measurements are taken. Metrologists discuss the details of measurement definition and validation. Process modellers discuss the ordering and organisation of development activities. These groups rarely meet, except perhaps in the field of software process improvement. It is the contention of this paper that software measurement and the software development process are (or should be) mutually dependent activities. The paper sets out to show these dependencies and examine their implications for organisations involved in software development.

Section 2 of this paper discusses the relationship between software measurement and the software development process. It first establishes the dependencies between measurement and the development process and then considers how the activities of the measurement process fit within the high-level stages of the development process.

One way of integrating measurement and process is through process modelling. An overview of process modelling is given in section 3. The uses of measurement within process models are also considered, as are the forms of process models suitable for measurement integration.

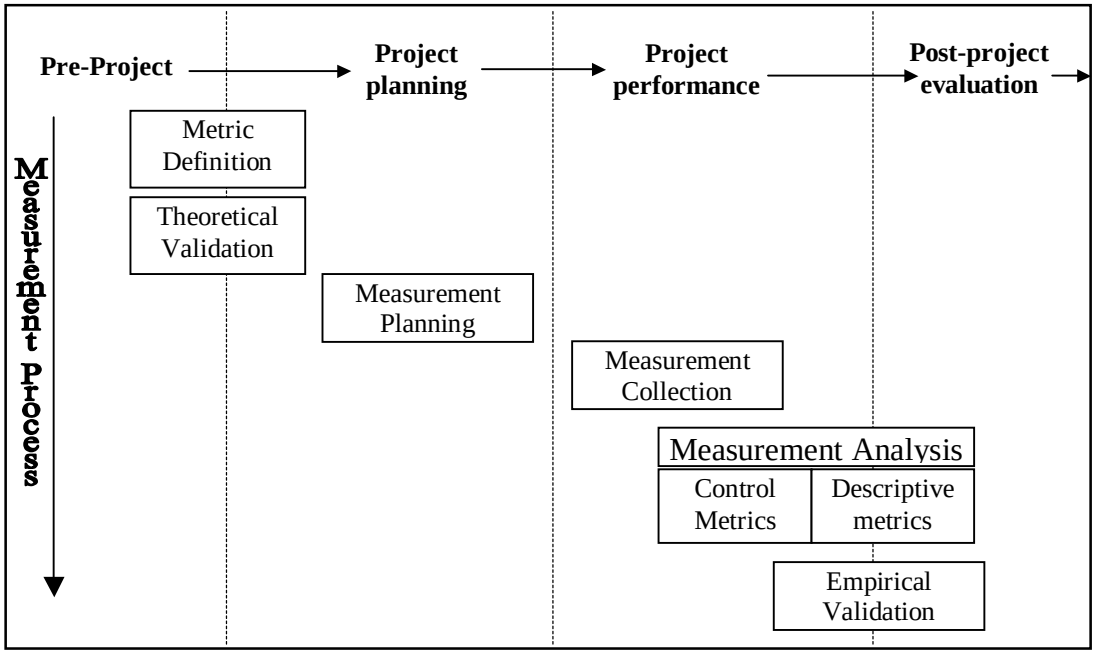
Software measurement is sometimes used for process improvement. The position of measurement within an improvement oriented organisational structure is also considered. This consideration is based around work done as part of the TAME project outlining an

organisational structure for process improvement combining support for model building, project guidance and measurement [1-3].

2. The relationship between measurement and process

The famous quote from DeMarco that "you can't control what you can't measure" [4] shows the dependency of development process control on software measurement. What may perhaps be less apparent, though equally true, is the dependency of measures on process. The needs of the process dictate what should be measured. The process defines when measurements should be taken and how the results of measurement will be interpreted (and perhaps feedback). For these reasons measurement should be considered as an integral part of the development process and not merely an adjunct to it. Figure 1 outlines the way that measurement process activities may be positioned within the high-level stages of a development project.

Figure 1: The position of measurement elements within project performance



When considering the integration of measurement into process, the first thing to consider is which elements of the measurement process require integration. At a basic level measurement programmes may be considered to consist of planning, collection, analysis and validation [5].

The first activities of the measurement process are metric definition and theoretical validation. Although these are necessary activities in an overall measurement *process*, they will usually have already been completed before the project process begins and so are not considered part of a measurement *programme*. They are included here mainly for reasons of completeness. However, it is possible that if a project has requirement for a new measure that these activities will be done during the project planning stage.

Deciding which measurements to make, and when to make them, ought to be done during the planning stages of the project and will be an integral part of the design/selection of the development process. It need not therefore be integrated into the development process model.

The collection and analysis of measures intended for project control must be integrated within the development process. Their use for process control can be integrated into the process model by using the results of measurement as event guards/triggers. This prevents activities from occurring unless particular measurement values (or ranges) have been achieved, or force activities to occur that would otherwise be bypassed. Within quantitative process models measurement values may also be used to select between alternative courses of action. Among other things, this provides a means of quantifying the exit criteria for iterative cycles within processes. However, care must be taken when using quantitative process models to ensure that they are not too prescriptive. The measures on which decisions are taken are often only indirect measures (or predictions) of the data which would ideally be used to make the decisions. Decisions based on such approximations should be made with a sensible degree of flexibility.

As well as control measures, descriptive measures may be collected. These are measures that do not directly affect the control of the project on which they are collected. Their use may be to help empirically validate metrics which are being considered for the control of future projects or to gain a quantitative understanding of what is happening within the project to help suggest process improvements. Adding this type of measurement produces what is termed an instrumented model. It is quite possible for such measures to be collected during a project but not analysed or applied until after the project's completion.

Empirical validation is an on going process, where each successful use of a metric increases confidence in its utility. The empirical validity of metrics may either, be reassessed as soon as the data becomes available (during the project), or may be left until after the project's completion.

3. Software process modelling

One obvious approach to integrating (or describing the integration) of measurement with the development process is with the use of process models.

3.1. What is Software Process Modelling?

Software process modelling is the formation of, and reasoning about, abstract representations of the processes and people concerned with the development and maintenance of software systems. The purpose of a process model will dictate the elements of the process that are included in the model. Curtis lists five possible purposes of process modelling: understanding and communication; process improvement; enhanced process status visibility; automated process guidance; process enactment [6]. Shepperd adds a sixth, the reuse of successful processes [5].

The particular aspects of a process that are modelled will depend on the detail of how it will be used. There are many aspects that may form part of a model. These aspects include:

1 the tasks to be carried out;
2 the order in which the tasks should be carried out;
3 the inputs and outputs to each task;
4 when and where a task will be performed;
5 how and why a task is done;
6 the roles and agents involved in performing the various tasks;
7 the iterative, interleaved and concurrent nature of tasks/sub-processes;
8 the conditions under which branches in process control are made;

9 the artefacts produced by or used in the process;
10 the resources used during a task;
11 the roles or activities which are dependent upon a task being completed.

3.2. Process Modelling Techniques

Shepperd divides the methods used in process modelling into five main categories [5]: lifecycle models; executable models; formal models; psychological models and graphical models.

3.2.1. Lifecycle Models

Lifecycle models are probably the best-known representations of software development processes. They are generally very coarse-grained models. Lifecycles can be split into two major types those derived for traditional structured development and those more recently proposed to allow for the differences in approach of object-oriented development.

3.2.2. Executable Models

Executable models are those which are written in computer enactable form. They tend to be much finer grained than lifecycle models. The advantage of an executable model is that they can be used to support automated software engineering environments. Several types of executable model have been used. These range from variations on standard procedural programming languages, such as the use of the Ada in the REBUS tool [7, 8] to functional languages and AI approaches.

This type of executable process model has been compared to software itself [9] . This has some interesting implications. If we can treat processes as software then the techniques and principles that can be applied to software production should also be applicable to process modelling. This includes: desirable properties like reversibility [10] and seamlessness [11]; constructing process models from reused components; and using process patterns for common types of modelling problems.

The above implications may only be considered useful or valid if we can model processes in the same way as software. As Curtis points out process modelling is different from other modelling in software engineering because some of the roles it models must be performed by people [6]. The introduction of people into the models adds a level of non-determinism which does not exist in software modelling [5]. Some might consider this to be sufficient grounds for treating software design and process modelling separately. However, software design techniques and notations are commonly used for representing process models.

3.2.3. Formal Models

Formal models involve the production of a mathematical description of the process. This type of process modelling has been done using both algebraic (OBJ) and model based (VDM) methods. Their strict formalism allows mathematical reasoning to be applied to process descriptions. However, formal method's determinism raises doubts as to whether they can properly describe human behaviour.

3.2.4. Psychological Models

Psychological modelling results in a very different type of model. They have their primary focus on modelling how people actually perform activities and make decisions. A model has been developed by Guindon and Curtis of designer behaviour using JSD [12]. This model shows that designer jumps between levels of abstraction and between design and

requirements whilst producing a solution. This shows a possible advantage of this type of modelling. Most process models of design would have assumed a textbook top-down or bottom-up approach. Without studying how people perform processes any process script may not be very precise.

3.2.5. Graphical models

Most process modelling methods fall into the graphical modelling type. This includes: general modelling methods such as flowcharts; notations borrowed from management planning like PERT charts or critical path analysis; software design notations from both structured methods (DFD, ERD, STD) and object-oriented methods (OMT, UML); and notations designed specifically for process modelling such as role activity diagrams (RADs).

3.3. Process modelling considerations

Deciding on the notation for a process model requires a clear purpose. There are a number of aspects of this purpose that may be considered to help make a choice of modelling notation.

3.3.1. Granularity

The granularity is the level of detail with which the process is described, or alternatively the size of the basic elements of the model. The purpose of a model will require certain information to be represented at a particular level of detail. If a notation is used which not capable of sufficient detail the purpose may not be achieved. If a notation is used which prescribes a finer granularity than necessary then the model will be cluttered with unnecessary information and consequently more difficult to use than it might be.

A related consideration in the choice of notation is the precision required of the model. The precision is the degree to which the model represents *all* steps to be carried-out to produce desired results. This is in effect the fidelity of the model.

If our purpose in modelling is to enable measurement to be integrated with the development process, then the granularity and precision of the method chosen must be commensurate with this purpose. Although lifecycles do provide a representation of the software development process, they have too coarse a grain to show how measurements would affect project control flow, or what measurements should be taken when. This leads to the conclusion that a more detailed description of software processes is required. Methods from within each of the other categories appear capable of representing processes at a sufficiently fine-grained manner to enable measurement integration.

3.3.2. Formality

The level of formality of modelling methods varies greatly from the strict formalism of VDM and OBJ to the more human centred psychological modelling. Process models intended for machine enactment are generally termed process programs. These models require a high degree of formality to give precise execution semantics. Process programs have the advantage that they are amenable to static checking (completeness, consistency and correctness) and dynamic checking (reachability, deadlocks and race conditions).

Process models that are intended for human enactment are usually termed process scripts. The non-determinism inherent in human action requires that process scripts have a substantial degree of flexibility within them. In general, for process descriptions intended for humans, expressiveness and comprehensibility are more important than strict formality.

If we again consider the various methods' suitability for measurement integration there are problems with the strictly formal methods like VDM and OBJ. The first problem is scalability. Although formal methods can be effectively used to represent fairly simple systems they may have difficulty with a fully formed industrial process. A second problem is the limited use and understanding of formal methods in industry and the associated reluctance to use them. These problems may mean that strict formal models are unsuitable for measurement integration on anything other than small academic examples.

On the other extreme, psychologically derived models may lack sufficient formalism to allow any useful computer enactment. It is generally agreed that industrial scale measurement will require automation. Possible difficulties in combining a manual process enactment with automated metrics collection may also rule-out psychological models for use in measurement integration.

3.3.3. Modelling style

Modelling styles can be divided into three perspectives, prescriptive, descriptive and proscriptive. Prescriptive models specify how the process must be performed. This detailed description of exactly what must be done can clash with human agents who tend to do things in a less regimented way. In such circumstances the process which is actually followed may often not be that which is prescribed. Descriptive modelling is used to determine the process that is actually being used. The third perspective on process modelling, proscriptive modelling, describes behaviour which is not allowed in the process. It is impossible in practice to describe the complete set of things which should not be done in the performance of a particular process (the list would be virtually endless). For this reason proscriptive modelling is usually used as an adjunct to prescriptive and descriptive models. In this situation they can be used as constraints on activities or events.

Since measurement may be used to describe processes or the control them via prescriptive or proscriptive means, any of these styles could be use for measurement integration.

4. Measurement and organisational structure

As was discussed in section 2, some aspects of software measurement involve inter-project activity that cannot be done within a single project's process model. Also, an individual project's process doesn't exist in isolation, it is designed and applied within the wider company structure and operates within an historical context of previous projects and the experience gleaned from them. To consider these inter-project questions an organisational structure will be examined that has been designed with these issues in mind.

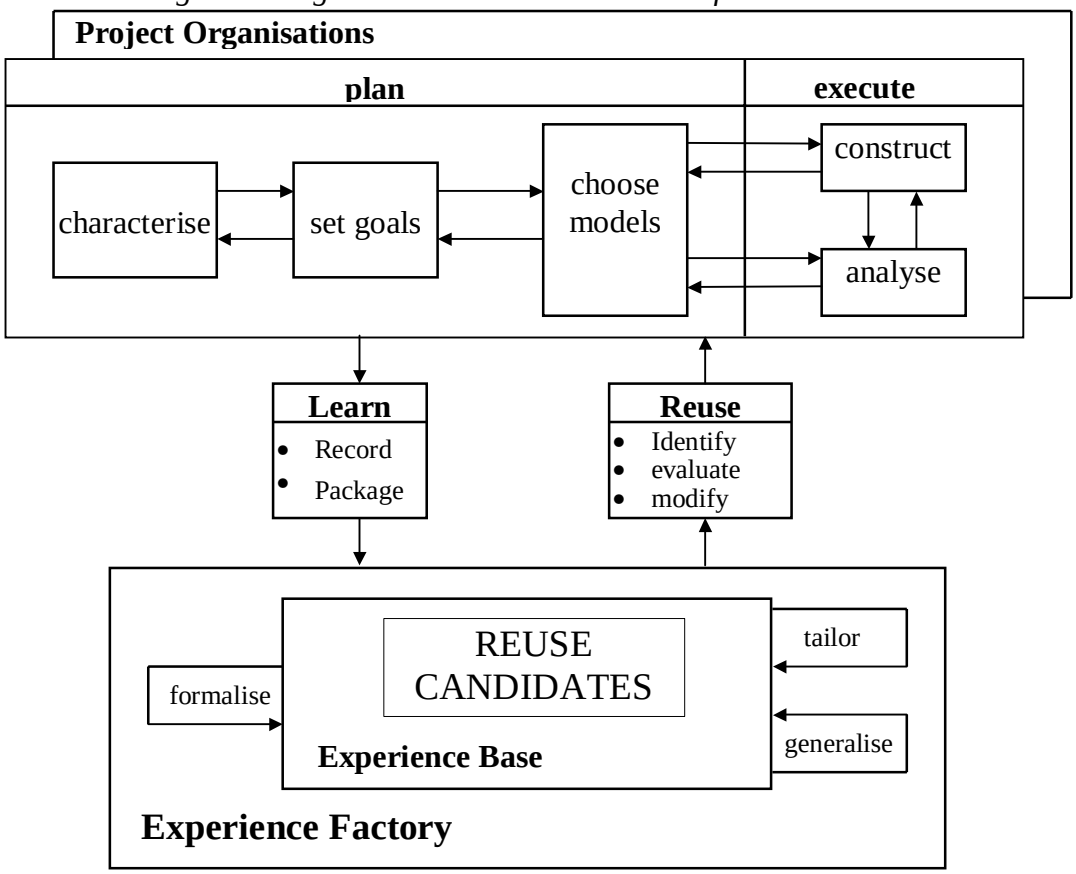
4.1. The TAME model

The diagram below shows the improvement oriented organisational structure developed by the TAME project [1-3]. The model is made up of individual project organisations which use (or reuse) elements supplied from an experience factory. The results from, and lessons learned whilst performing, each individual project are used to populate the experience base.

An individual project has its own characteristics and goals. These are used to choose appropriate models and metrics from the experience factory to be used during the project performance. The models may include product models, process models, resource models, and quality models. The project is then planned, performed and monitored using the models and metrics. Records of the project performance and any lessons learned during the project

are fed back into the experience factory. This new experience is then packaged for future use, or used in the repackaging of existing models and metrics.

Figure 2: Organisational structure tailored for measurement



4.1.1. Project organisations

A project organisation deals with the performance of a single project. Within each project there are a set of standard activities, i.e., characterisation, goal setting, model selection, construction, and analysis.

Characterisation takes place during a project's planning stage. It involves assessing various project attributes in order to aid model selection. In order to be useful there must be a record of the characterisations of previous projects, the various models they used and the degree to which those models were considered to fit that project after-the-fact.

Goal setting is the selection, and operational definition, of goals for the construction and review (analysis) of the system being built. The team who produced the TAME model were also central in producing the goal question metric paradigm (GQM) [13] and they suggest relating the setting of project goals to measurable outcomes using GQM.

Model selection uses the analogies drawn from the characterisation to chose models to be used in the performance of the current project. As has been previously mentioned, it may be that there is no sufficiently close model to use and a new model may need to be constructed.

Construction is the process of actually producing the end product of the project. The construction is done according to the chosen development process model and the selected methods and tools.

The analysis and packaging of data from a project organisation must be done with reference to the existing data in the experience factory. Using knowledge of the existing models and metrics it must be decided what form of analysis would produce results suitable for packaging, storage and reuse.

4.1.2. Experience factory

The experience base holds the complete body of experience that is available for use by the project organisations. The experience factory contains the experience base and also associated processes for working with stored experience. These process include: tailoring existing experience to be reused in new contexts; generalising specific experiences to make then more widely applicable (and hence more reusable) and formalising experience into models that may also be reused more easily.

In general, the experience-base may contain any product or process entities that a company feels may be usefully reused. If a more specific view is taken, relevant to integrating measurement with process, the set of entities of direct interest can be listed and discussed. These entities may include entity models such as attribute models, process models, resource models, quality models and process models. They may also include the goals and questions used to select measures, the measures themselves, measurement results and the heuristics or thresholds used for the interpretation of measurement results.

4.2. TAME and the measurement process

4.2.1. How is measurement used in the TAME model?

There are several places within the organisational structure where measurement is used. Measures may be used to characterise projects. Project goals should be set in measurable terms (often via GQM). The models used in the construction and analysis phases of a project, e.g., quality models, are often measurement-based. As Basili states, 'analysis (including measurement) cannot be an add-on but must be part of the execution process and drive the construction' [1]. This means putting then actual construction of the product under measurement control.

4.2.2. How does the TAME model facilitate measurement?

A major advantage to measurement of using the TAME model would be in planning a measurement programme. Measurement planning requires the selection of which measures to collect as well as how and when to collect them. An experience base can help with this is several ways. It can provide process models (or model sections) already packaged with the necessary control measures, including when and how they should be collected and how the results will be used. Storing historical GQM hierarchies can help to associate a new project's goals with relevant measures. Pre-packaged quality, resource or product models may also be defined in terms of measures. All of this stored experience can be used to reduce the effort in measurement planning.

As mentioned previously, empirical validation is an on going process, where each successful use of a metric increases confidence in its utility. This means that the validity of measures must be constantly reassessed each time new data becomes available from a project.

The database of measurement results must be maintained outside of any individual project, as it will contain the results from many projects.

The storage of historical measurement data would also be useful in the tuning of measurement thresholds or ranges used in process control or quality evaluation. Again this data is collected across a number of projects and so must be stored outside of any individual project organisation.

5. Summary

Software measurement and the software development process are (should be) mutually dependent. The most promising means for integrating measurement and process seems to be process modelling. Process models intended for measurement integration should be sufficiently fine grained to show when, and how measurements should be taken, who should take then, and how they will be used. The necessity for this level of granularity excludes the use of the standard lifecycle models.

Measurement can be most easily integrated within an automated environment as the measurement collection must also be automated. This may lead to a preference for executable process models for measurement integration. These executable models may, however, have problems representing non-deterministic human behaviour. This problem of ease of automation versus ease of understandability and precise representation of human behaviour doesn't appear a particularly tractable one. It seems likely that trade-off and compromise are required here rather than attempting to find an 'ideal' solution.

A measurement programme may be split into its constituent parts of planning, collection, analysis and validation [5]. Of these parts the measurement planning must be done during the project planning stage before the development process model can be enacted. Validation may be split into theoretical validation and empirical validation. Of these theoretical validation may be done during project planning. Measurement collection must be integrated with the development process and taken into consideration when choosing/designing the process. The analysis of any measures that can effect the control of the process must also be integrated with the process. The analysis of descriptive measures that do not impact process control may be done during the process or may be left until after the development is complete. Empirical validation may be done as soon as the results are available, but may often be left until after the completion of the development.

Empirical measurement validation and measurement reuse are mutually dependent activities. Unless a measure is reused the necessary corroborative evidence cannot be gathered for validation and assessment of generality. Unless a measure is validated it will not be reused. Both of these aspects of software measurement involve inter-project activity that cannot be done within individual project organisations. To consider these inter-project questions we need to define the organisational structure within which individual projects take place.

One model for an organisational structure is that produced by the TAME project. This structure is used as an example to demonstrate the interaction of measurement and process and show the facilities that need to be in place, at an organisational level, to aid measurement integration with process. It also shows how these facilities interact with individual projects.

6. Conclusions

If measurement and process are mutually dependent they should be planned, stored and reused together. Measurement programmes require cross-project structures for data storage, analysis, threshold tuning and validation. The TAME model provides the mechanism for achieving these goals.

Would the author recommend every software development company to drop their existing organisational structure and adopt TAME? In reality, no. No one would seriously consider swapping their organisational structure in a revolutionary way. So you might say, what's the point of the work? The TAME model provides an example of an organisational structure with facilities to support measurement and for development to be supported by measurement. It isn't necessary that the TAME model be adopted. However, it is desirable is that those in software development gain an understanding of the inter-dependencies of measurement and process, and understand the facilities that need to be put in place in order to take advantage of them.

7. References

[1] Basili, V.R. and Rombach, H.D., "The TAME project: Towards Improvement-oriented software environments", IEEE Transactions on Software Engineering, 1988. 14(6), p. 758-771.

[2] Lott, C.M. and Rombach, H.D., "Measurement-based guidance of software projects using project plans. Information and software technology", 1993. 35(6/7), p. 407-419.

[3] Basili, V.R. and Rombach, H.D., "Support for comprehensive reuse", report No. CS-TR-2606, 1991, University of Maryland.

[4] DeMarco, T., "Controlling software projects", Yourdon Press, New York, 1982.

[5] Shepperd, M., "Foundations of software measurement", Prentice Hall, 1995.

[6] Curtis, B., Kellner,M.I. and Over, J., "Process modeling", Communications of the ACM, 1992. 35(9), p. 75-90.

[7] Sutton, S.M., Heimbigner, D. and Osterweil, L.J., "Language constructs for managing change in process-centered environments", ACM SIGSOFT Sofware Engineering Notes, 1990. 15(6), p. 206-217.

[8] Sutton, S.M., et al. "Programming a software requirements specification process", in 1st IEEE International Conference on the Software Process, Redondo Beach CA, 1991.

[9] Osterweil, L.J., "Software Processes are Software Too", in 9th International Software Engineering Conference, IEEE Computer Society Press, 1987.

[10] Shepperd, M.J. and Ince, D.C., "Derivation and validation of software metrics", Open University Press, 1993.

[11] Avotins, J. "Towards an object-oriented metric modeling method", in OOPSLA'96 - workshop: OO product metrics, 1996.

[12] Guindon, R. and Curtis, B., "Control of Cognitive Processes during Design : What tools would support software designers?", in Proceedings of CHI'88, New York: ACM, 1988.

[13] Basili, V., Caldiera, G. and Rombach, H.D., "The goal question metric approach", in Encyclopedia of software engineering, Wiley, 1994.