

A process modelling approach to managing software process improvement

D.S. Hinley, K.H. Bennet

Centre for Software Maintenance, School of Engineering & Computer Science, University of Durham, DH1 3LE, UK

ABSTRACT

A Software Process Model (SPM) need not simply be regarded as a static, descriptive representation of the software process with similar strengths and weaknesses to those described for traditional Software Development Life-Cycle (SDLC) models. Research is currently being carried out to study how SPM's may be of direct benefit to software managers to sustain or improve software quality during the maintenance process.

By focussing on the management aspects rather than the technical activities undertaken, the changing roles of the software manager and the problems currently faced in managing complex processes can be highlighted. Some of the ways in which SPM's can be developed and used by managers to gain greater insight and understanding of the software process have also been elucidated.

This paper outlines how SPM's may be used in the analysis of existing processes and how they may contribute towards the design and implementation of new and improved processes, in order to better achieve managerial objectives. Initial findings, within the framework of our experimental investigation, suggest that process modelling is a useful technique for establishing and developing **roles**, with the result that a positive effect on the co-ordination of resources and in the development of group and individual relationships could be qualitatively measured. Skill levels and communication pathways have also seen improvements based on the enaction of prescriptive models.

190 Software Quality Management

INTRODUCTION

Software Engineering has been recognised as an engineering discipline for almost 25 years. The term was originally promulgated at the International Garmisch Conference in 1968, and was applied primarily to computer programming activities. It was originally perceived as encompassing a variety of techniques and tools which were considered to facilitate the production of cost-effective, reliable software within specified time constraints.

Since 1968, the discipline has matured slowly with the growing realization that software engineering is not solely concerned with computer programming, it also embodies managerial, social and economic considerations. These concerns are reflected in a recent definition by McDermid [1], who defines software engineering as:

'the science and art of specifying, designing, implementing and evolving - with economy, timeliness and elegance - programs, documentation and operating procedures whereby computers can be made useful to man.'

The broadening of our understanding of this field has in itself introduced software management difficulties, which will be discussed later. However, the growing acceptance that engineering-style disciplines can be applied to the software process, has led to the consideration of the work processes involved and the necessary roles and skill requirements for practitioners. Furthermore, as process details are elucidated, attention can be given to the methods employed and the techniques and tools which may improve software quality and productivity.

An important principle, often over-looked during method evaluation or whilst procuring new techniques or tools, is that a method should follow a strategy for achieving the desired software state. It is desirable that the strategy provides an approach that conforms to a particular process model. The results of not applying this principle are frequently seen e.g. incompatible methods, instability of method used, often resulting in frequent changes before their effect has been evaluated. Under these conditions much software development and maintenance may be regarded as chaotic, even though some entry level quality technology may actually be applied e.g. product inspection and reviews.

The often expensive mistakes of the 80's and the economies of the 90's have focused both business and software managers' attention on the core business and its effectiveness. Systems and Computing

departments are no longer regarded as simply an exceptional overhead and they must address their business performance. Even where integrated methods and CASE tools have successfully been introduced their resultant contribution e.g. in terms of productivity and quality to application development and maintenance is broadly unquantified, despite the claims of major vendors.

The profile of IT managers is now higher in many organizations; often with Executive presence, they need to increasingly justify their position and allay fears that software development and maintenance cannot be effectively managed. This is an increasing challenge as both the size of the IT budget and the increasing dependency of an enterprise on its computer-based systems are a growing, if not altogether managed risk.

Managers have relied on models which provide an abstract representation of the complexities of the problem area or dilemmas faced in the real-world. A typical model familiar to most software managers is the Project Plan, which is usually devised to represent the activities which are necessary to provide a solution to the engineering problem. The plan may often be a specific instantiation of a more generic model for a software project e.g. the Waterfall or 'V' - model. These models provide a much needed structure, allowing attention to be focused on the key development stages and may be used to communicate about and achieve a mutual understanding of the structure of a software project.

Generally software development and a substantial proportion of software maintenance has traditionally been considered in terms of a project model. The concept of a project may not always be compatible with our understanding of the software life-cycle, particularly when considering quality issues including cost. This can result in some inconsistency which inevitably leads to management difficulties e.g. conflicts between the roles of quality and project management. A typical problem which can arise during development is the extent to which maintainability or reusability should be considered. This is an important issue; if software is to be considered from the quality perspective, a long-term management viewpoint is required i.e. throughout the software life-cycle. However, such considerations would appear low on the project development manager's agenda; particularly as project deadlines and budgets are more immediately and directly measured in comparison with quality attributes.

To focus on the software process and its relation to other processes, both internal and external to an organization, may offer a solution not

192 Software Quality Management

only to such typical management dilemmas as cost, time, quality issues, but may also provide a suitable mechanism by which software process quality improvements may be synchronized with broader business direction and objectives.

The remaining sections of the paper provide relevant background to a project which is providing a practical insight into how models may be used to manage process change in line with business objectives.

PROJECT BACKGROUND

Our current research interest in SPM's and how they may be used to at least sustain or improve quality during software maintenance, is being conducted as an integral part of an IEATP project 'Models and Metrics'. This collaborative project involving Ferranti International and Lloyd's Register of Shipping, provides a suitable industrial framework in which new models may be applied to the management of software projects and especially software maintenance. The framework of the project also provides an assessment mechanism, based on both qualitative analysis and quantitative metrics, through which the affect of process change may be measured.

During the course of the project we have endeavoured to create a 'best-practice' model of maintenance (BPM), and the trials assessment of a number of process elements is being fed-back into our modelling in order to evolve the BPM through furthering our knowledge and understanding of suitable practices. It was acknowledged that such a generic model would be difficult to implement into existing project practice; furthermore, it was not immediately evident what the real benefits that even a detailed model could provide, from the software management perspective.

A number of uses have been put forward e.g. it could be used as a datum for formulating a suitable questionnaire to ascertain the level of software process maturity evident on existing projects; furthermore it may be used to assess the level of quality technology across projects and also the suitability of management metrics associated with software maintenance activities. Our current research project is likely to explore further these areas in the near future, particularly now that there is growing commercial and industrial interest in software process assessment, and that management and quality metrics are essential for measurable process improvement.

Before proceeding with a more technical consideration of software process improvement and our management approach it is useful to

consider some development aspects of a software process model.

Software process model development

A SPM has been defined by Dowson and Wiledon [2], as:

‘a purely descriptive representation of the software process, representing attributes of a range of particular software processes and being sufficiently specific to allow reasoning about them’.

This general definition provides little guidance on what may constitute a suitable model and perhaps has inevitably led to much confusion and misuse of the term; however, the 4th International Software Process Workshop [3], did attempt to list the constructs and objects, as well as suggest some of the capabilities which a model should demonstrate.

Currently there is no single universally recognised notation for a software process model. A number of generally recognised formalisms have been used by researchers to demonstrate various model capabilities, including: project planning, resource scheduling and cost estimation.

Software process model notation

Before beginning our practical model building, a brief survey of other known process model notations was undertaken. These included process programming languages e.g. PML [4], analysis and design paradigms, plans, ETVX activity diagrams, functional and behavioural approaches [5], rule-based approaches and role activity diagrams. Our study indicated that modellers had generally chosen to adopt and sometimes adapt a convenient notation, based on the readily available CASE tools or support environment utilities, rather than on stated criteria and their satisfaction.

We were not constrained by the use of particular tools, so chose to consider the purpose and the use of our models, as the underlying basis for our modelling effort. We concluded that any resultant SPM's should be responsive to practitioner: (software engineer and manager), demands e.g. measurable improvement to specified quality attributes; they should represent organizational requirements i.e. how the business relates to the software maintenance process, represent the roles which are enacted throughout the process, underpin process improvement initiatives and be capable of being evolved as our understanding of the software process and its limitations grows.

From the constructional viewpoint we also assessed the recognized constructs, objects and capabilities suggested for SPM's and

194 Software Quality Management

chose to represent: process, product, information flow, agents, roles, and constraints. The model architecture was essentially hierarchical, representing different perspectives, process sequence and some control aspects.

A key feature of our SPM's is the recognition of **roles** i.e. the logical grouping of human actions to satisfy specific process elements. This was found to have many advantages in both the creation of models to describe existing practice and in comparing different models and prescribing new or improved processes in model terms. Such improvement models could be readily accepted by practitioners, even though new responsibilities and communication networks were being established. The practitioners' understanding of the model was reinforced and amplified by their preparation of detailed procedures. Indeed the ability to reason about the process at the process planning stage resulted in much iterative improvement, before implementation. Ownership of the process was unambiguously transferred from modeller to practitioner, as the process was prescribed in their terms, without the human resource problems which sometimes ensue if an external agent considers jobs, work breakdown and post hierarchies.

SOFTWARE PROCESS IMPROVEMENT

Software engineering by definition represents a number of discrete processes throughout the software life-cycle. Our research has focused on the **software maintenance** process; by software maintenance, we broadly adopt the IEEE definition [6], given below:

'modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment.'

Of particular importance is the inclusion of both functional enhancement to meet changing business requirements and the transformation of existing software to ensure that a system remains operational when the hardware or the operational infrastructure changes. This somewhat global view of software maintenance establishes its primary importance in any software process improvement initiative, particularly if the chronological aspects of the software life-cycle are considered. Furthermore, it may be instrumental in the feedback of pertinent information for the continual improvement of the software development process e.g. to provide insight for both software engineers and managers as to which product attributes can facilitate and improve software maintenance.

Process improvement in engineering is well established, particularly in the domain of manufacturing. The evolution of product engineering design and manufacture has not simply been limited to the technical aspects of product innovation, but has included the development of new organisational and management perspectives. The focus of process improvement has changed from simply seeking direct increases in productivity towards dynamic processes, capable of being customised to suit specific customer requirements. This external customer focus is exemplified by a genuine concern with product quality and through the evolution of more sophisticated quality technology. A basic quality principle is now established within manufacture engineering that the processes undertaken dictate the product quality attributes. This principle has not yet been generally accepted by the software fraternity, but there is growing evidence of the importance that software procurers now place on the software process e.g. awareness of the applicability of Quality Management Systems (QMS) to software, and the recent interest in process assessment (Software Engineering Institute (SEI) - Process maturity model, Esprit - Bootstrap assessments).

It is generally accepted that in order to improve any process, a base-line must be established i.e. the existing process needs to be described and assessed. This may lead to a definition of the existing process, which can then be evaluated with respect to some idealised model or particular goals.

A process improvement framework is often prescribed by which process improvements may be proposed and planned for integration, to augment, replace, or generally modify existing practice. There are a number of existing frameworks which are based on the familiar Plan/ Do/ Check/ Address (PDCA) - type cycle. This process improvement mechanism does however make a number of fundamental assumptions which may not be valid for the software process under investigation; for example:

- the process is understood;
- the actual process is established;
- the cycle-time is considerably shorter than the life expectancy of the process or its derivatives;
- measurements of both the process and the products are defined and agreed.

Some doubt may therefore exist on the immediate benefits which can be achieved by applying generally accepted engineering frameworks and TQM paradigms to the software process. The invalidity of the above assumptions for the software process is substantiated by existing software

196 Software Quality Management

process assessments already undertaken by both the SEI and the Esprit Bootstrap Consortium, which indicate that only a small percentage of assessed organizations have a defined process.

It may be argued that establishing the existing base-line by means of a recognised process assessment method may reduce some of these problems, and may enable software process improvement initiatives to follow. However, a preliminary evaluation of the merits and weaknesses of existing software process assessment approaches, concluded that the balance was heavily influenced in favour of seeking a new paradigm, for the direct assessment of industrial-based projects and the software maintenance process which is actually enacted within a specific commercial or industrial domain. Our underlying assumption is that an optimum software process is both dependent upon the business objectives and the prevalent circumstances i.e. external environmental factors e.g. customer or market constraints.

PROCESS MODELLING APPROACH

In the previous section, we have outlined our concerns with respect to two notions: the establishment of the existing process base-line using a direct questionnaire-based assessment method, and secondly, attempting to use general TQM or other process improvement paradigms directly for software process improvement.

Within the scope of our research we are actively seeking to suggest and perform some trials evaluation of different approaches and incrementally develop both a real software process and our approach to software process improvement. Our focus on software maintenance has been beneficial in providing further insights into the role of both the software engineer and manager; this has led to the early recognition of a significant shift of emphasis within the traditional software management role. We currently perceive software managers to be focused not on project (product-based) management, but more acutely on two aspects: managing the software process and managing process change.

It has been recognized for some time that the software process is inherently complex and difficult to manage. These difficulties are often centred on planning, monitoring and controlling the process. Perhaps the most significant contributory factor is the tenuous transformations of task deliverables throughout the process and the high reliance on human endeavour and creative ability to understand and define the requirements, and produce and test the appropriate software.

Advances in software engineering have tended to focus on the technical aspects of the process. There is a proliferation of methods, techniques and tools to support a range of technical processes. Success with the available portfolio of engineering aids has been tempered by both the increasing demand for larger, flexible systems and the higher regard for the resultant product quality as perceived by the customer.

To support the management of development processes geared towards producing large, yet flexible systems, quickly and with due regard to quality and budget, new management approaches have been sought which recognize the project management risks. Experiences have later been reflected in general models e.g. Boehm's Spiral Model [7], for the benefit of other managers. The Spiral Model suggests an incremental development approach, recognizing the need to plan and re-evaluate, based on the consideration of the risks emerging during the course of the development. Unfortunately, such models do not provide much support for identifying management measures or dynamically reflecting actual processes. Furthermore, many of the software development models have poor representation of maintenance activities; we believe however, that the management dilemmas outlined above, and the poor representation of software maintenance may be specifically addressed through an SPM.

General framework

We have already discussed our intention to develop a BPM for software maintenance during the course of our research project. The initial BPM was established based on the results of a study of existing maintenance models, referential knowledge within the Centre for Software Maintenance and empirical studies. The resultant model is necessarily a generalized abstract form and has been constructed in a hierarchical fashion. It does however, represent more than simply a maintenance stage breakdown, prescribing the technical process and the managerial activities and management control mechanisms: organizational, request, change, and release.

A key factor in undertaking modelling work is in defining the scope of the model. This was facilitated by our initial decision to adopt a standard definition for software maintenance to form our process boundary. An investigation of published maintenance models, indicated that the majority failed to represent the management perspective and provided little support for process management e.g. they did not reflect many of the immediate needs faced by managers in aligning software processes with the general business, and to be proactive in the development of both process and product. Increased accountability and responsibility for an important value-added element

198 Software Quality Management

of the business or product is now resulting in the need to produce detailed information on existing capability and limitations. A new process boundary was described by means of a Context Model [8] thus establishing a business process viewpoint of software maintenance.

Having established the scope of the BPM, a high level model was produced to represent activity domains and their related generic prerequisite deliverables, products and information flows. Figure 1 presents a portion of the high level model relevant to the improvement areas chosen for experimental trials.

Functional decomposition of the activity domains resulted in a logical model containing 90 process steps for software maintenance; the DFD's were then structured to reflect a number of participant viewpoints. We were particularly keen to embody the notion of a QMS as well as more technical management control mechanisms.

We feel that the resultant BPM is useful in its own right, in terms of addressing the general lack of understanding of both managers and customers, of the software maintenance process. Furthermore, the BPM may prove useful in supporting the management issues described above. For example, our organizational control model provides some insight into the managerial processes and potential data collection sources for effective and rationale forecasting, resource management and strategic development.

IMPROVEMENT SCOPE

Our approach to defining the scope of process improvement is outlined in this section and further details of our initial trials on an industrial-based software process indicates some of the benefits we have found through establishing existing process base-lines, and assessing the defined process against management objectives and our BPM.

A model of current practice (CPM), was originally derived from the results of a hierarchical task analysis of activities performed by the trials project team. The descriptive model was reviewed and the resulting dialogue allowed more detailed communication aspects of the process to be gathered.

To facilitate in the definition of the process improvement scope, the managerial goals were established through direct discussions with a cross-section of managers. Our involvement in this exercise was to act as a catalyst, to ensure all levels of management opinion were in fact discussed, and to facilitate in defining managerial objectives which were

supportive of the overall business goals. The team effort resulted in the identification of six goals which were then ranked in order of importance to the business. For each goal the critical success factors i.e. the process elements which significantly affect the achievement of the goals were also ascertained and ranked. For example, changes in the general business climate had necessitated a change in maintenance project costing philosophy, towards fixed-price maintenance projects. It was therefore imperative that projects effectively operated within the agreed budget envelope. Critical factors affecting a project's ability to operate within cost included:

- early establishment of the requirements and the scheduled deliverables;
- the effectiveness of the request and change control mechanisms;
- the number of changes to the requirements specification and the phase at which they arise;
- the effect of changes on the defined technical and managerial processes;
- ascertaining the relevant factors for estimating;
- integrating estimating with project planning and customer liaison;
- incrementally improving estimates and reacting through each project phase;
- monitoring actual progress, comparing actuals and estimates, and realizing the consequences of plan deviations;
- assigning people and optimizing the use of facilities for each task;
- assessing training, experience and skill requirements for each task;
- evaluating software complexity and group productivity.

The process improvement scope was determined by defining areas of the existing process which contributed towards poor achievement of managerial objectives and which were likely to contribute towards the problems outlined during the analysis. The CPM and BPM were compared to specifically identify which existing process elements could be replaced, amended or augmented to likely result in better achievement of managerial objectives and minimize the existing reported problems.

From the overlay or mapping of the two models several issues emerged which were of interest to the maintenance project team. For example, the existing practice was indicative that management did not

200 Software Quality Management

perceive that they had much involvement with strategic issues. Prototype technology also was not influential on existing practice; data collection and the use of quality records as a basis for management review and decision support was not widespread or undertaken on a rational basis. These weaker process areas were though to contribute towards some of the reported product assurance dilemmas relating to project management.

The direct comparison of models enabled a large number of process improvement ideas to be put forward, of which the following were selected by the software project for enactment:

- the clarification of rules and roles for the estimation process; it was recognised that a high degree of flexibility was required, but detailed guidelines on the project and phase specific factors would be beneficial;
- the inclusion of an explicit risk management process for the project;
- managerial forecasting e.g. technical time distribution based on activity profiles, to aid in predicting bottlenecks and over-runs;
- management analysis of on-going situations through demonstrative planning and control e.g. action lists, estimates, target/actual comparisons, justified plan changes.

A logical model of the selected process improvements was drawn to indicate interfaces with existing processes, and to determine a suitable implementation strategy. This model was beneficial in providing a suitable transition plan from which the processes to be enacted could be specified in detail. Furthermore, the improvement process could be refined e.g. the existing project close-down activity was modified to consider project success criteria, and process and product data evaluation and archiving.

PRELIMINARY ASSESSMENT

A qualitative analysis using a questionnaire which had been formulated from a slight variation of the Goal / Question / Metric paradigm [8], was used to assess the achievement of managerial goals on both a previous major enhancement and the existing modification being undertaken on the system. The qualitative assessment has only allowed us to consider the package of process changes rather than the effects of individual process element changes. However, in this particular experimental trial, the results have been encouraging, with

the cost goal being adequately achieved and some marginal positive effects on other goals. Of particular importance is the recognised improvement in team awareness and orientation towards the business domain.

The modelling activity has been beneficial in a number of other ways. For example, the BPM representation of a generic best-practice found favour with the organization for a variety of reasons:

- it was non-intrusive, it could be looked at from a variety of perspectives and compared to existing practice;
- it graphically represented both technical and managerial processes, products, data-stores, information flows and roles. This helped practitioners to envisage the overall process and their role and prompted detail discussions on how they may develop their roles and improve the existing process;
- The designation of data-stores provided management with further insights into defining useful process and product metrics which may provide further characterisation of quality attributes and their ability to control the process.

Our approach to establish the existing process base-line by means of a graphical model i.e. defining a CPM, also found favour with the industrial trials project. The approach overcame some of the weaknesses of a rigid Questionnaire, in that it did not assume from the outset that there was a standard means of assessing process maturity - an emotive concept especially when being applied by an external agent. The subsequent assessment of the existing process by comparing the CPM and BPM, with respect to the management objectives, critical success factors and known process problems, ensured that process improvements were tailored and prioritized to the real needs of the organization at that point in time.

A comparison of the CPM and BPM resulted in approximately 30 substantive improvement ideas. The graphical depiction of the improvement scope, facilitated management dialogue on the appropriateness and relative merits of the improvement ideas, and when final selections had been made, also enabled a transition plan to be formulated. This is a key deliverable in change management and in process improvement planning, and enables new management metrics to be ascertained for monitoring the effects of the process change in advance, so that data-collection may be synchronised with process change.

202 Software Quality Management

CONCLUSION

Descriptive models of existing software processes using systems analysis techniques can be formulated relatively easily. To develop these logical models into a suitable SPM for considering areas of process improvement requires a slightly different approach. Our trials have suggested that role orientated analysis is useful in establishing the necessary communication and behavioural perspectives of a software process.

Best-practice software processes are thought to be business, application domain and environment specific. However, we have shown that a strong process definition and abstract model can be developed using functional decomposition to produce a prescriptive generic model. The CPM and BPM may be compared to assess the existing process base-line with respect to its ability to satisfy management objectives. The model-based assessment provides a suitable framework for describing, defining and planning incremental process improvements to the existing base-line, in order to satisfy managerial goals and reduce process problems.

ACKNOWLEDGEMENT

We wish to acknowledge the work of all of the members of the IEATP project team: 'Models and Metrics', for their contribution to the success and applicability of the Process Modelling work being undertaken at Durham, and the funding of this Research Project by both the DTI and SERC.

REFERENCES

1. McDermid, J.A. (Ed). *Software Engineering Reference Book*, Butterworth - Heinemann, 1991.
2. Dowson, M. and Wileden, J.C. 'A Brief Report on the International Workshop on the Software Process and Software Environment' *ACM Software Engineering Notes*, Vol.10, pp.19-23, 1985.
3. Tully, C. (Ed). *Representing and Enacting the Software Process*, Proc. 4th. Int. Software Process Workshop, Moretonhampstead, Devon, ACM Press, 1989.
4. Roberts, C. 'Describing and Acting Process Models with PML,' in *ACM SIGSOFT Engineering Notes*, Vol.14, No.4, 1989 (Ed. Tully, C.) pp.136 - 141, *Proc. 4th. Int. Software Process Workshop*,



Moretonhampstead, Devon, ACM Press, 1989.

5. Kellner, M.I. 'Software Process Modeling: Value and Experience' *SEI Technical Review* - 1989, pp.23 - 54, 1989.
6. ANSI/IEEE 729: *Standard Glossary of Software Engineering Terminology*, IEEE, 1983.
7. Boehm, B.W. 'A Spiral Model of Software Development and Enhancement,' *IEEE Computer*, pp.61 - 72, 1988.
8. Hinley, D.S. and Bennett, K.H. 'Developing a Model to Manage the Software Maintenance Process' to be published in *Proceedings of a Conference on Software Maintenance*, Orlando, Florida, 1992. IEEE Computer Society, 1992
9. Rombach, H.D. and Basili, V.R. 'Quantitative Assessment of Maintenance, an Industrial Case Study,' pp.135 - 147. *Proceedings of a Conference on Software Maintenance*, Austin, Texas, 1987. IEEE Computer Society, 1987.

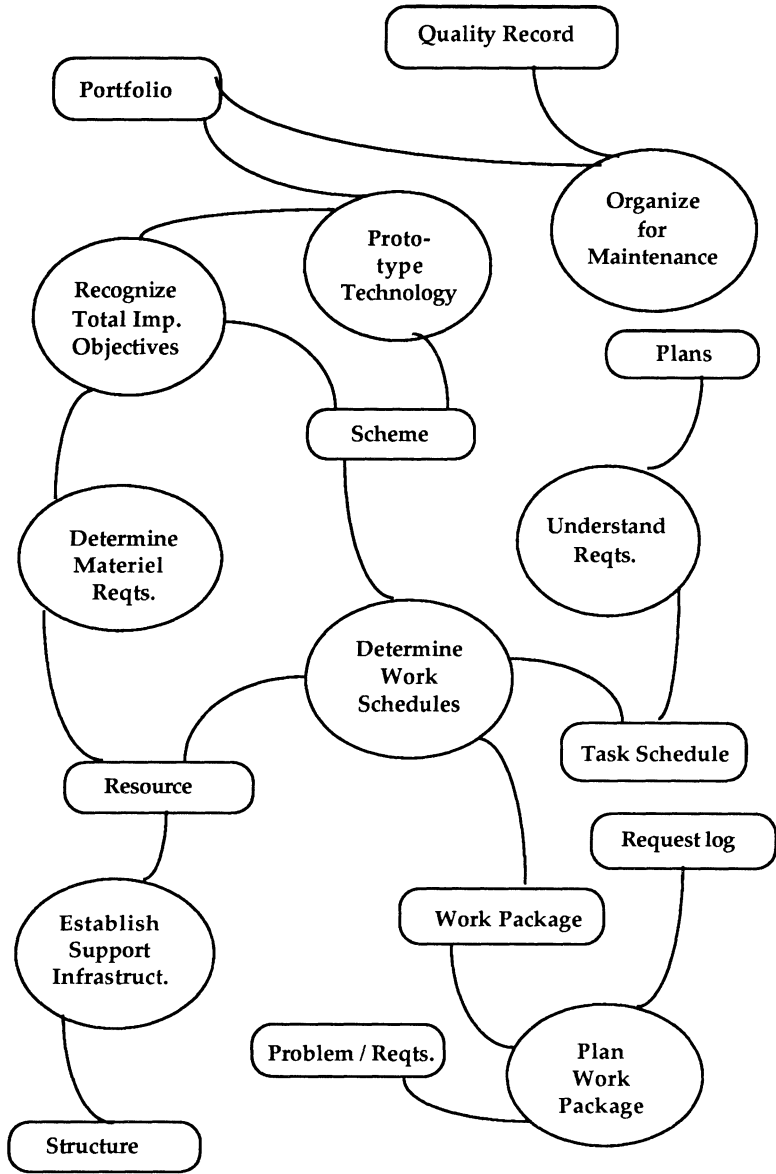


Fig. 1 Representative Sample of High Level Model