Computer Physics Communications 41 (1986) 217–225
North-Holland, Amsterdam

PROJECT MANAGEMENT IN THE DEVELOPMENT OF SCIENTIFIC SOFTWARE

Jochen PLATZ

Siemens AG, 8000 Munich 83, Fed. Rep. Germany

This contribution is a rough outline of a comprehensive project management model for the development of software for scientific applications. The model was tested in the unique environment of the Siemens AG Corporate Research and Technology Division. Its focal points are
- the structuring of project content – the so-called phase organization,
- the project organization and
- the planning model used, and its particular applicability to innovative projects.

The outline focuses largely on actual project management aspects rather than associated software engineering measures.

1. Goals of the project management model

The overall objective of the project management model is the further enhancement of performance and efficiency in software development.

Items intended:

- Improvement of software product quality,
- Shorter software project lead times,
- Timely elaboration of secured statements on project results, time frames and costs,
- Optimal utilization of available resources.

For the realisation of the model special consideration is taken for:

- the acceptance of these methods by highly qualified scientists,
- the support of the sociotechnical introductory tasks to complex projects by these methods.

The aggregate of these objectives has led to an overall project management concept with particular emphasis on the characteristics of software development. This concept emphasizes the synergy of all methods rather than the effects of individual methods.

2. Environment of the project management model

The subject model was developed in a multiproject environment. The applying department unit is structured by subject expertise; the majority of projects handled boast a very high degree of innovation. For this reason, a concrete definition of tasks and a basic solution path can frequently not be determined at the starting of the project. They have to be elaborated "as the project goes". Yet a meaningful model application lends itself only to tasks permitting an early-phase definition of ultimate development goals. Activities to which this model is applied are designated projects.

The content and economic feasibility of project goals are continually defined within the project scope. In this way, the model supports the differential process of problem definition, as well as the final determination of performance scope and solution approach. The scope of projects handled with this model varies from case to case, starting with applications involving at least 4 scientists and gaining full applicability for larger projects enlisting the service of more than 20 scientists.

The project requirement catalog is continuously rising. In an environment of increasing sophistication and complexity of subject content, innovation-oriented topics call for ever shorter completion deadlines, while the complexity of results, expressed in software component numbers and their interdependence, is likewise rising.

3. The project management model

3.1. Basic organization

In the model presented, the individual project management methods have been reconciled in such a way that they can be harmonized and combined into a "Joint Project Management System". The methods are linked by a precise and forward-looking definition of each formal result with respect not only to the actual software development process, but also to ancillary processes such as development of tests, documentation and planning details. The channeling of all methods within the model also determines the structuring of individual methods.

The following five key elements form the core of the integrated model (see fig. 1):

1. A process of *project goal definition*, with special emphasis on innovative elements.
2. A logical approach geared to problem definition. *Procedural organization*, phase organization.
3. A *project organization* tailored to software development criteria.
4. Operative *planning* adapted to innovative philosophies.
5. Project execution follow-up and *control* adjusted to applicable lead times.

Aside from these key or core elements, the model encompasses a large variety of other integrated elements, such as:

- Order management,
- Requirements management,
- Structure planning,
- Configuration management,
- Quality assurance,
- Reporting,
- Decision organization.

To achieve a uniform and efficient model, all methods have been based on the following six key principles:

1. Result-oriented performance,
2. Personalized responsibility for component results,
3. Rigid structuring of results and procedures,
4. Step-by-step decisions and completion,
5. Team-oriented definition of objectives,
6. Support of ingenuity and creativity by admission of latitude.

3.2. Procedural organization

The procedural organization segments the approach in logically completed development steps or phases. The decision on execution or nonexecution of a given project is made in an initial plan-

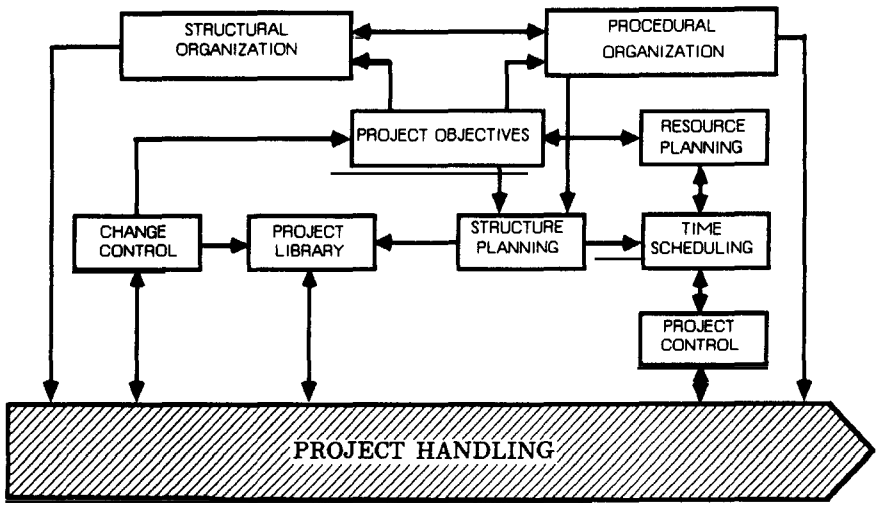

Fig. 1. Key elements of project management.

ning procedure. A project run on this model conventionally starts with a development request, which must not necessarily be comletely defined as to content. It ends with the acceptance of the software product either by the requesting department unit or by the user. If no concrete requesting unit can be pinpointed, the completion of work may be determined by explicit decision.

In its distinction as a phased process, the procedural organization encompasses the following characteristics:

1. The project execution period is subdivided into individual phases, which are end-pegged with milestones defining component results. The milestones form a logically constructive chain of results; the sequence of questions or approaches attaching to the project is as follows (see fig. 2).
   More often than not, a clear delimitation of problem and solution areas cannot be attained. The model, therefore, provides for overlap phases and reentry into completed phases. The control of these processes is handled via configuration management.

2. The development process is split up into several independent and unrelated partial processes which run separately but parallel or in a division-of-labor mode, as the case may be. Each partial process is again subdivided into phases which correspond to the overall phase model. At each phase-end the results of the partial processes are reconciled; the reconciled results are referred to as baselines. The principal partial processes are:

   - the software development process, comprising
     - task clarification,
     - solution search,
     - realization;
   - the software engineering process;
   - the testing process;
   - the planning process.

3. For each phase of each partial process the result to be achieved will be defined at start of project in the form of a generically neutral description with detailed itemization. The software engineering methods and tools to be used are important components. The starting point is a more general description which is to be modified in accord with actuals during the following phases. In conjunction with the subsequent time frame information, the project execution milestones are determined via the results, such as the project requirement catalog, the project-related software engineering guidelines, all studies and preliminary surveys, all draft documents including product and test specifications, the planning documentation and,

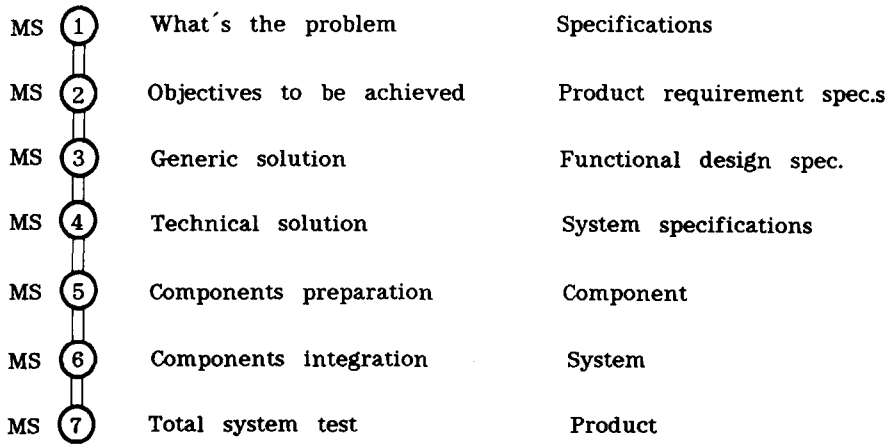

Fig. 2. Chain of questions and results.

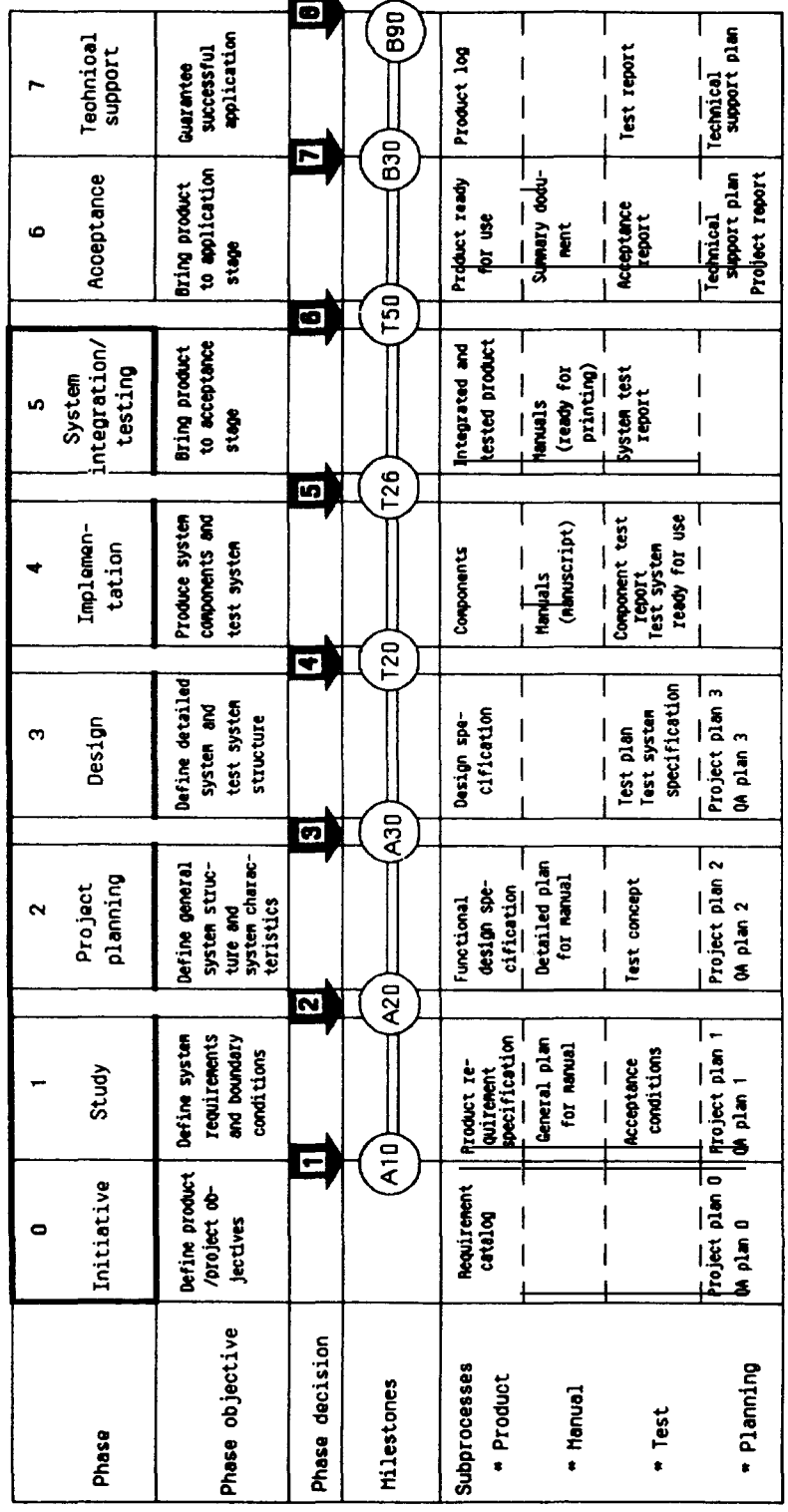

Fig. 3. Phase organization.

last but not least, the software components already elaborated.
In view of the high innovation content, the result planning activities must provide for ample latitude and intuitive working climate. This also necessitates continuous balancing between creative judgment and strict application of available software engineering measures. For this reason, partial results emerging in the process of task clarification and solution search should not be pegged before it is ascertained that their realization will be feasible during the subsequent phases. On the other hand, a straight sequentialization of partial processes must be avoided, especially in view of the desired short lead times. Fig. 3 shows the phase organization summary sheet.

4. Each single result is subject to continuous quality assurance evaluation over the entire project span, which is effectuated in successive steps. The initial step is a check of the result achieved versus input standards and project-related definition of results. A subsequent step determines whether or not the given single result fits in with the results that were achieved in parallel or previous efforts, while the evaluation whether and to what extent the result content corresponds to actual and topical requirements is reserved to a third subsequent step. These evaluative steps are supported by special methods based on group dynamic processes (walkthrough, review)and by checklist controls. The overriding thought behind these efforts is the recognition that software quality is established during the early stages of development and that quality can only be "designed into" rather than "tested into" the result.

5. All milestone results are inspected and accepted by decision in accordance with a defined procedure. This will assure that the phase results of the individual partial processes are collected in reconciled fashion. The phase decision simultaneously elucidates the technological, scientific and economic risks inherent in the project.

6. The results that are passed are being incorporated into the project library, while anticipated results are preidentified and all relevant documentation and software components are managed on dp equipment. It is further assured

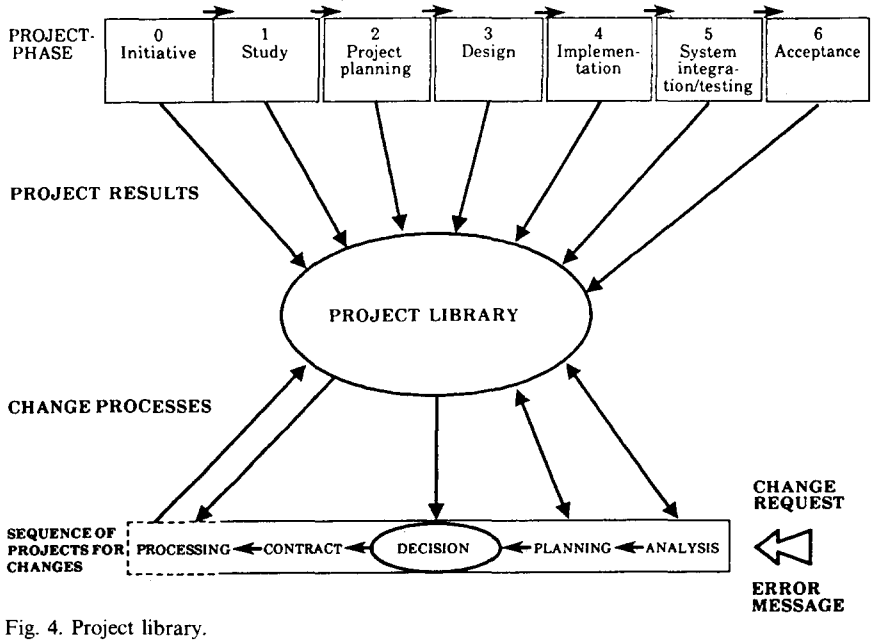

Fig. 4. Project library.

that changes to passed development results, which may be triggered by change requests, problem reports or test reports, will only be implemented via clearly defined change procedures and following a decision by a configuration management Change Control Board.
The data stored in the project library are the binding basis for all subsequent work and the total project (see fig. 4).

3.3. Project organization

The project organization is of prime importance. The organization model presupposes a certain number of project functions to be performed for every software project, independent of the people in charge and the organization pattern involved. Fig. 5 illustrates the structure of the corresponding functional model.

The project functions are subdivided into subfunctions with work packages keyed to the project structure plan. The attached project functions:

1. Project management. Responsible for the organization of the project and its environmental integration; further for project target definition, personnel recruitment, training guidance and related functions.

2. System engineering. Responsible for the planning of system content (requirement engineering), definition of development process content, adaptation to phase organization and software engineering methods, definition of interfaces between the system to be developed and its environment, and between system components and other relevant items.

3. Development. Responsible for the elaboration of system components as defined by system engineering; draft documentation, software components, such as models, procedures and file organization; user manuals for component testing.

4. System integration and test. Responsible for assembling the prepared system components and modules, and subsystems for total system; activation of test strategy, test tools and test data; performance of the system tests.

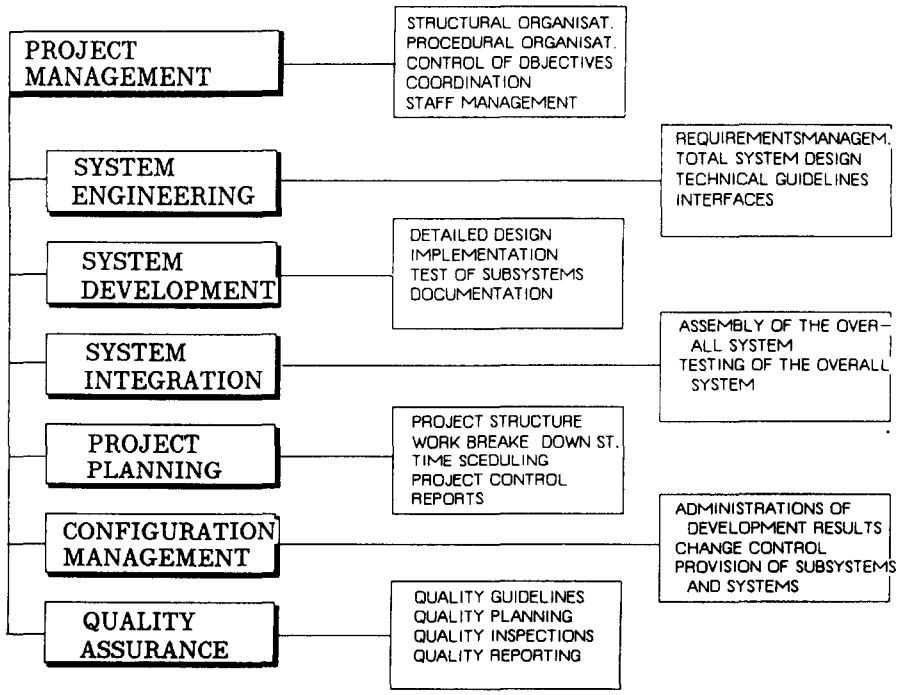

Fig. 5. Structure of functions.

5. Project planning and control. Responsible for the operative planning of the project, reporting, implementation of control measures, project management, etc.

6. Configuration management. Responsible for management of project results, documentation and components; handling of all change transactions within the project.

7. Quality assurance. Responsible for testing intermediate and final results for required quality characteristics.

8. Project organizational structure The functions and the functional scope that will be needed for a given project are determined via the basic model of project functions. The weight and content attaching to individual project functions may be shifted during the various project phases.

The project-specific organization with responsibilites and competence delegated to project units is set up on the basis of the project functions and structural organization as outlined above. This approach enables the formation of optimal organizational patterns for every project on hand. Individual functions, such as quality assurance, configuration management or system integration and testing may be administered by central units for several projects. A detailed description of process steps and operations explains and regulates the functional interaction within the project.

3.4. Operative planning

The project planning function aims at determining project processing costs and deadlines as timely and as accurately as possible. Problems may arise in software development under uncertainty, especially with regard to

- objectives which still have to be clearly defined,
- feasibilities of request realization, subject to checks during the life of the project,
- realization paths to be elaborated "as the project goes".

The planning model provides for two separate planning echelons with graduated range and accuracy of plans.

In the so-called project plan, the total duration of the project is considered under specified uncertainties. The project plan is successively updated in accordance with the attained level of current knowledge. The processing of phases is organized in the so-called phase plans. The key data for deadlines and costs for the various phases are adopted from the project planning function. As the phase planning focuses on a self-contained and logical segment of the project, it can as a rule be performed with a sufficient degree of accuracy.

The actual planning procedure is subdivided into a string of separate and upgraded planning steps, with each step focusing on only one single planning aspect. In this way, the complex generical and logical continuity of operative planning is translated into unequivocal and controllable steps. Fig. 6 illustrates the sequence of these steps.

1. The *project structure* segments the desired software product into hierarchical single components. The software product requirements

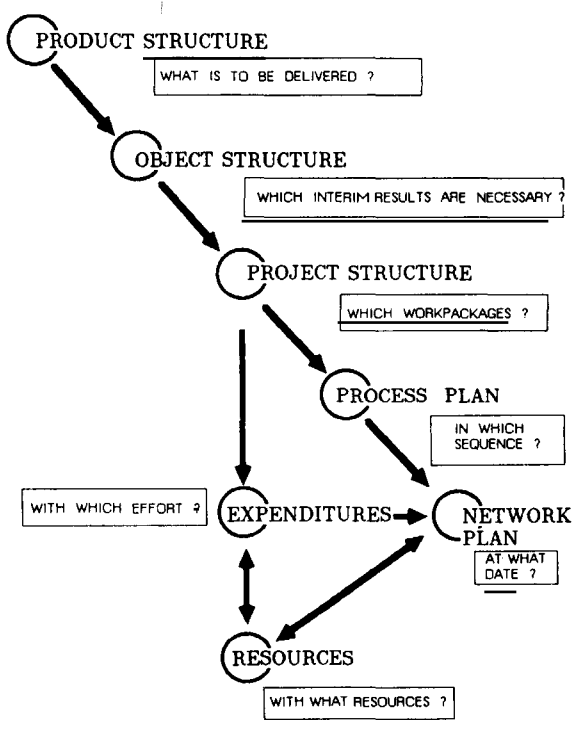

Fig. 6. Sequence of planning steps.

are the basic criterion of project structure. The question is: What are the concrete results and performance features the project is to yield?

2. The *object structure* itemizes phase results of subprocesses and other subresults. This plan determines which cost-relevant objects the project is to elaborate over its total duration. In addition to product structure, attention is focused on phase organization and the software engineering process.
3. The *project structure* shows all work packages that are necessary for the preparation of each object within the object structure. The project structure plan is the actual central plan which coordinates all planning and control activities of the project. Support functions are the object structure and the project-specific functional model.
4. The *process plan* provides the generic and logical sequencing and intermeshing of work packages of the project structure plan. Influencing factors are the software engineering and phase organization functions.
5. The expenditures accrued per work package of the project structure plan are recorded in the *expenditure budget.* Determinatiion of expenditures may become problematic for software projects when developments with innovative content are concerned. Expenditure estimates for this model are based on individual estimates for each work package of the project structure plan. This very differentiated estimation philosophy allows for flagging of critical or disorganized work packages which are then segregated from the normal routine packages. The marginal conditions of project processing may be applied direct to the pertinent work packages, and following the keying of deadlines to the work packages, the budget will indicate expenditures by period and further structured by project functions. This facilitates a very efficient planning of resources.
6. The *network plan* establishes the resource-related deadlines for work packages. This plan evolves from the process plan, following determination of lead times for each work package on the basis of available capacities and accrued expenditures.

If a project calls for it, a number of subplans must be derived from the master plans, e.g.:

- a quality assurance plan,
- a personnel placement plan,
- a risk hedging plan,
- a subsupply plan.

The influence of uncertainties may become problematic in the planning of innovative software products. This is why the planning model provides for hedging measures and procedures to precisely collect borderline condition data, incorporate all changes and new findings, and feed back new results.

4. Model introduction

The success of project management methods is often thwarted by lackluster introductory effort, particularly when such methods are launched in complex organizational environments. In a similar way, this also applies to the application of software engineering methods. The introductory impact may on the other hand be notably enhanced and supported by intelligent method conveyance, as exemplified by the following system in the form of a method module kit:

- The methods can be separately applied and incrementally integrated.
- The methods correspond to terminology and philosophy of the applying unit.
- The individual method has been formulated as simply and intelligibly as possible.
- The method surface structures can be adapted to project-specific criteria.

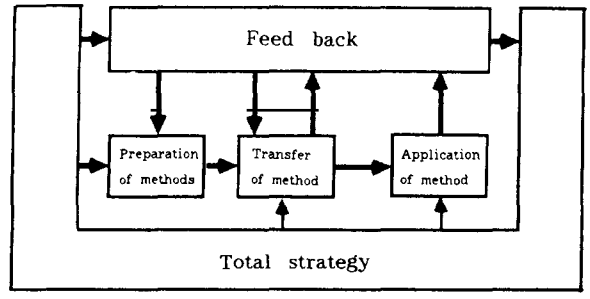

Fig. 7. Model of method introduction.

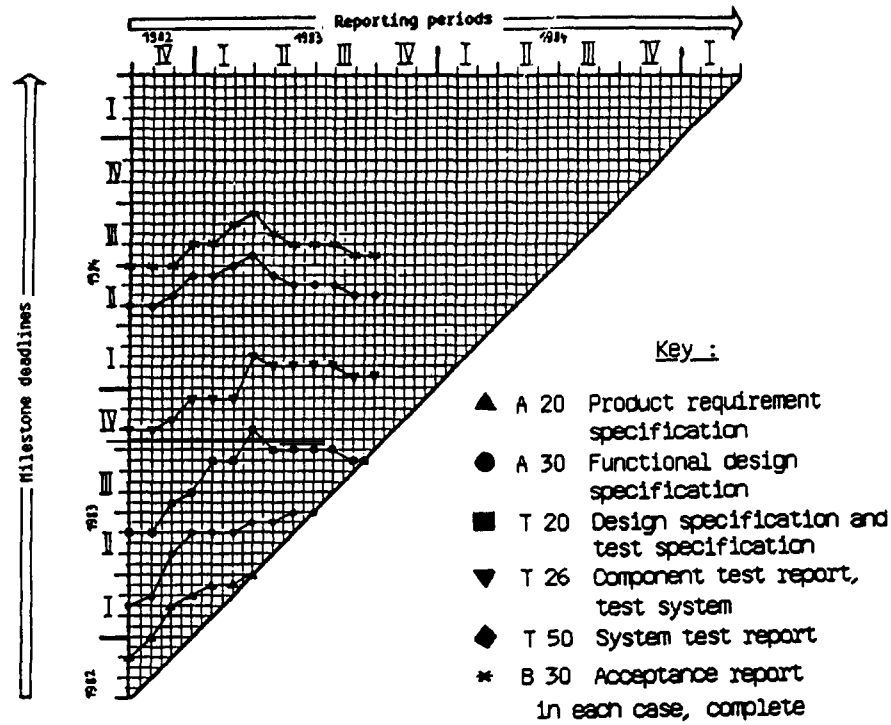

Fig. 8. Milestone trend analysis.

The model features a user-oriented method introduction approach (see fig. 7).

In this context, a horizontal training effort was launched for project managers, rank-and-file staff and line managers. Individual method segments, such as milestone trend analysis (see fig. 8), have been introduced to the entire echelon of qualified staff. New projects are handled from the outset in full accordance with the model systematism. Various stages of introductory coverage have currently been achieved for longer established projects; for some projects the introduction phase is all but completed. We are happy to report that practical experience with this model has gained positive and favorable acceptance with project managers, staff and management.

References

[1] I. Albert and J. Platz, in: Projektmanagement, Beiträge zur Jahrestagung 1983 (Munich, 1983).

[2] E.H. Bersoff, V.D. Henderson and S.G. Siegel, Software Configuration Management, An Investment in Product Integrity (Prentice Hall, Englewood Cliffs, 1980).

[3] B.W. Boehm, Software Engineering Economics (Prentice Hall, New York, 1982).

[4] P. Bruce and S. Pederson, The Software Development Project (John Wiley, Chichester, 1983).

[5] W. End, H. Gotthardt and R. Winkelmann, Software Development (John Wiley, Chichester, 1983).

[6] K. Gewald, G. Haake and W. Pfadler, Software Engineering (Siemens Aktiengesellschaft, Berlin, Munich, 1977).

[7] J. Platz, in: Projektmanagement, Beiträge zur Jahrestagung 1984 (Munich, 1984).

[8] R.H. MacDonald, in: Project Management, Tools and Visions, Proc. 7th Inernet World Congr. (1982).

[9] B.J. Madauss, Projektmanagement (Pöschel-Verlag, Stuttgart, 1984).

[10] Ph.W. Metzger, Managing a Programming Project (Prentice Hall, Englewood Cliffs, 1973).

[11] M. Saynisch, in: Projektmanagement. Konzepte, Verfahren, Anwendungen, eds. M. Saynisch, H. Schelle and H. Schub (Oldenbourg, Munich–Vienna, 1979).