The Project Matrix:
A Model for Software Engineering Project Management

Sally Shlaer
Diana Grand
Stephen J. Mellor

Project Technology, Inc.
10940 Bigge Street
San Leandro, California 94577-1123
510 567-0255
http://www.projtech.com

Abstract

The Project Matrix is a project management model of a software development project. This model requires no special resources other than those normally assigned to a software development project and has proved to be effective in coordinating the work of many people, managing the operations of the project, reducing the complexity of the software development process, and producing high quality results.

Keywords: software engineering project management, project management, project planning, requirements tracing, project control.

1. Introduction

This paper reports on a simple model for technical project management, the Project Matrix. We have found this model to be effective in coordinating the work of many people, managing the operation of a software development project, reducing the complexity of the development process and producing high quality software. Further, this approach requires no special resources other than those normally assigned to a software development project, and can therefore be operated on a single-project scale, needing no particular institutional structure (such as a Technical Writing or Quality Assurance Department). Finally, the model is easy to explain and simple to grasp—it has intuitive appeal for people of a technical orientation.

2. The Project Matrix Model

2.1 The Matrix Proper

The Project Matrix is a model of the software development work of a project. In this model, the work is represented by a matrix of activities, such as that shown in Figure 1. Each row of the matrix represents a type of task (for example, writing specifications, developing an information model, coding, etc.) and each column represents a "subsystem" for which the task must be performed. Each box on the matrix therefore represents an activity, preferably of a few weeks duration, for which there is a well-defined output object: a requirements document, a set of structure charts, a collection of code, or the like.

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

	1: Timing	2: Data Acquisition	3: Setpoint Control	4: Diagnostics	5: Dosimetry	6: Exposure Regulation	7: Patient Record Management
1: Construct Information Model							
2: Develop Requirements							
3: Develop External Specification							
4: Design Programs							
5: Design Data Structures							
6: Code							
7: Test							

Figure 1: A sample Project Matrix

As the activities of the matrix are undertaken, the output objects, or "documents", are produced. All of the documents related to a single subsystem—a column—are collected together and ordered like the rows of the matrix to which they correspond. This ordered collection of documents results in a Subsystem Notebook, with the rows of the matrix providing a standard organization for all of the notebooks.

By means of the Matrix and Subsystem Notebook concepts, the work of the project has been partitioned into a clearly defined set of deliverables whose nature reflects both the structure of the application being addressed and the methodology selected for the work.* The development staff sees the work as an orderly filling-in of the various Subsystem Notebooks.

The project manager's problem, then, is to develop a matrix which lays out the work in terms of the objects which the project staff is required to produce—either for internal use in software development ("intermediate deliverables") or for external delivery to the client.

2.2 Information Dependencies Between Activities

The development of the matrix must take into account the information required for production of each successive box. These dependencies are depicted in an information flow diagram which applies separately to each column of the matrix. Figure 2 shows the information flow diagram associated with the matrix presented earlier, wherein the activities (the rows) are now represented as bubbles, and the output objects as labeled arcs.

*The Subsystem Notebooks are based on thinking very similar to that represented by the Progressive Project Documents[1] and share many of the same properties. A large difference is one of packaging: The PPDs are somewhat similar to row-wise collection for the matrix documents.

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

Some points to note:

- The information flow diagram represents fundamental information flow constraints of the development work.
- The diagram brings out the possibilities for parallel work within a column.
- The diagram does not illuminate the information flow constraints which may exist between tasks in separate columns.

Information flow constraints provide a basis for sequencing work. This is discussed more fully in Section 6 under Project Scheduling.

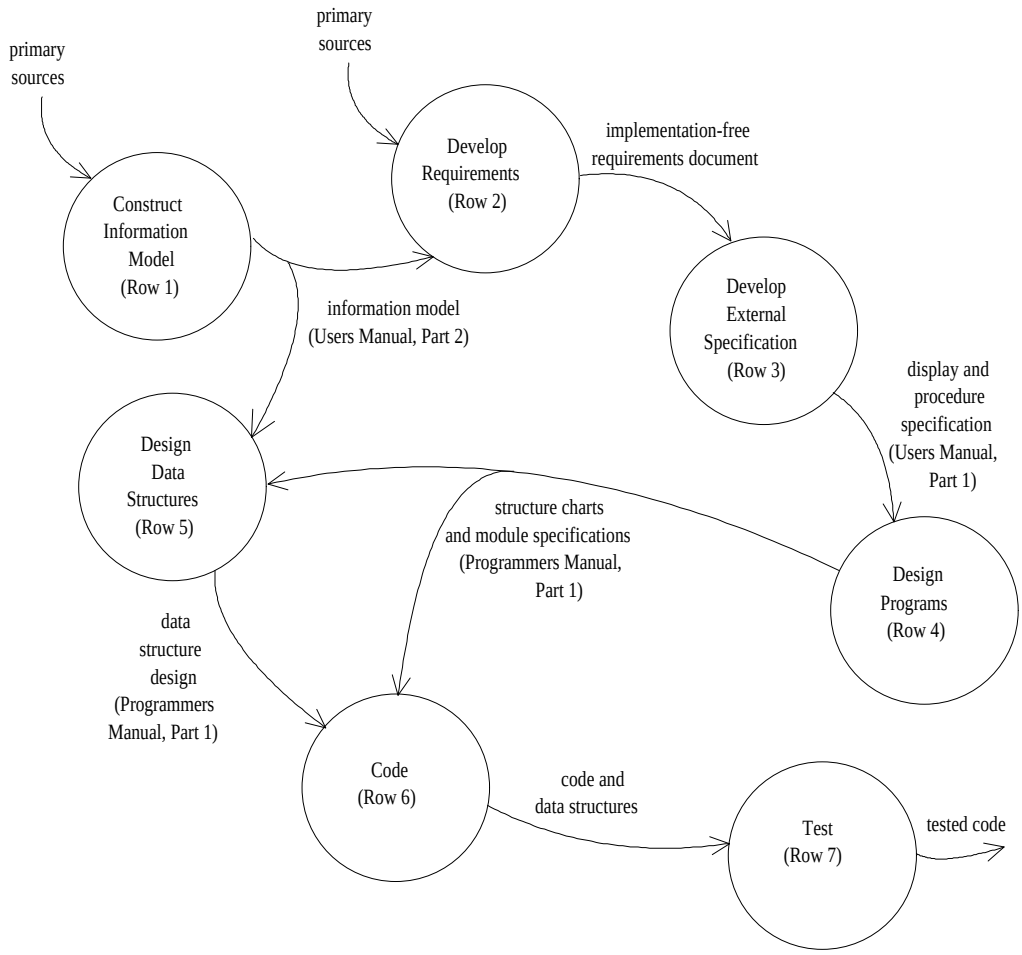

Figure 2: Sample Information Flow Diagram

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

2.3 Subsystem Notebooks

We have found it advantageous to develop a document numbering scheme which ties the output objects directly to the matrix from which they were generated and to the Subsystem Notebooks for which they are intended. The following form for document identifiers provides such a scheme:

mm.xx.y.z

where

mm denotes the matrix being referred to. A short alphabetic identifier is appropriate here.

xx is the number of the Subsystem Notebook. It is the same as the number assigned to the matrix column, as shown in the annotations of the column headings in Figure 1.

y is the chapter number within the specified notebook. It is the same as the row number in the matrix which is to generate the material for the chapter.

z is a mini-document number within the chapter (row). This allows for the generation of a number of small documents within a given box. The orientation towards small documents is of significant help in getting the work out[1].

3. Multiple Applications

The matrix is an integrating tool that provides a single framework for organizing, relating and viewing the following somewhat diverse aspects of the project:

- the work breakdown (the tasks that must be done to complete the project)
- the on-paper deliverables (e.g., documents, drawings, listings, etc.)
- the computer files: their organization is readily based on the column structure
- project staffing: the assignments of people to tasks, the quantity and types of skills required
- the current status of the project (i.e., how much work is completed)
- the project's software engineering standards and methodology

Equally important, the Project Matrix has a significant impact within the technical domain. The effect and value of the row partitioning so placed upon the work is well understood[2]. In addition, the column partitioning acts to keep separate technical problems well separated through all stages of development. This is a divide and conquer approach, and the benefit is a reduction of complexity in the delivered system.

4. Communicating the Project Methodology

Key to the success of the Project Matrix approach is the firm definition of the required output objects. The construction of an additional notebook, labeled “Project Methodology”, has proven to be a useful technique. The methodology notebook is organized with one chapter corresponding to each matrix row.

Each chapter contains the software engineering standards applicable to the row. The standards include:

- content and intent: What information is to be stated for the row and why
- form: Conventions for the expression of the information
- quality criteria for acceptance of the work

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

Having this material in a convenient form for reference is helpful in ensuring that the methodology is consistently observed; further, having it identified with the project (and so closely linked to the project's work) may promote its internal acceptance, in that it is less likely to be viewed as an arbitrary imposition from the outside.

The tone of the Methodology Notebook directly affects how the project members think about the work. A book which focuses on prescription of form encourages the staff to adopt a mechanical approach to the work. A formalism-based approach tends to promote a research atmosphere and direction of thinking. We have had best experience with methodological exposition which highlights the purpose and intent of the particular step being described, emphasizes the information content required in each row's output objects, and which deemphasizes descriptions of form.

With or without a Methodology Notebook (not all of our projects have elected to produce such a document), the project's methodology is most effectively communicated by example. It is therefore a good strategy to produce the work of a single, preferably simple, column early. This serves to check out the methodology in depth and, together with the matrix itself, makes it clear to the developers just what is wanted for filling out the Subsystem Notebooks.

5. Considerations for Matrix Construction

Defining Columns. The goal is to define a set of columns (subsystems) which can be developed relatively independently of one another. A software engineering approach to this problem is, first, to develop an implementation-free model[3] of the system and, secondly, to partition this model to produce subsystems with the desired property. The drawback to this approach is that it defers use of the matrix until a substantial part of a “breadth first” analysis is complete. Given the time involved in establishing such a model and the system requirements, this may result in a considerable delay.

A second approach, which has been surprisingly successful, is to partition the problem into subsystems intuitively, drawing on the experience of specialists well-versed in the application area. This provides a first cut which is “approximately right”, and vastly facilitates the work required to get the partitioning established on a stronger basis. Our experience is that this strategy may require that adjustments be made to the matrix after the project is underway, but that these adjustments are small and easily accomplished.

Defining Rows. The goal is to define a set of rows which serve as well-placed stepping stones to delivery of the running system. This is accomplished by focusing on the intermediate deliverables produced by each row and arranging for each successive deliverable to capture an appropriate and traceable increment of additional development work. An ideal set of rows will produce a reviewable expression of the work at each significant turning point in the development path.

Exactly what that reviewable expression of work is to be is given by the row definition in terms of the software engineering techniques and formalisms to be used. In selecting these formalisms, the project manager will want to consider the tradeoffs between the learning curves associated with unfamiliar tools and the increased effectiveness such a new tool might bring to the project. It has been our experience that a few new techniques can be morale-boosting by their professional development effects, but too many new ideas are destabilizing.

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

It is a good idea to examine candidate tools for their likelihood of clashing with previous experience. For example, we have found data flow diagrams frequently ineffective for process control engineers and electronic designers: These disciplines depend on techniques which compound the concepts of data and control—a forbidden practice in data flow construction.

Use of Multiple Matrices. Each matrix represents a specific methodology defined by its set of rows. Several matrices may be required on a single project, particularly if system software or software architecture work is required as a part of the project in addition to the application itself. Also, if a project contains some work which is being done from scratch along with some work which is heavily based on previous code, different matrices with different row structures may provide the best support.

Number of Boxes. Because there seems to be a fixed cost for "getting in and out of a box", one should try to keep the number of boxes from becoming too large. In addition, it is useful to associate internal project reviews with the completion of each box. Too many boxes will discourage this practice. These observations need to be balanced with the benefits of having a work breakdown made up of many small tasks.

6. Lessons Learned In Use of the Matrix

Project Scheduling. The boxes of the matrix provide elements which can be used to develop the project schedule, perhaps with assistance of a computerized critical path management scheduling tool. Scheduling can also be done on a coarser grain, using a chain of boxes as a schedulable task. The information flow diagram supplies a set of such chains for each column together with their interactions within the column. Interactions between columns, as well as dependencies on external events (delivery of hardware, say), are not exposed by the matrix or the information flow diagram, and must be added independently. However, since the effect of the column partitioning is to make the number and intensity of these interactions small, the problem of stating these interactions is substantially eased.

Visibility of Work. The Project Matrix approach makes all phases of the software development work unusually visible, including those not directly associated with code production. While ordinarily perceived as an advantage, there is a down side. For example, we have heard comments like "When are you going to complete this 'paper mill' phase and get on with the real work?" Since the matrix approach assumes that the paper is the work—or the externally observable and reviewable evidence thereof—an analogy with the design and review procedures of other engineering disciplines can help to explain what is really happening on the project.

Creativity. The matrix supplies tremendous amounts of structure—sometimes far more than the technical staff is accustomed to. Some project members may feel that their creativity is being unnecessarily limited. For this reason, it is better to err on the side of underprescription in the Methodology Notebook, and to remain sensitive to this issue.

Staffing Strategies. The project matrix makes it straightforward for the project manager to weigh alternative staffing strategies. Because the project is partitioned into small cohesive tasks, the quantity and type of expertise required can be easily seen. This facilitates assignments of people with specific talents and skill levels to appropriate tasks.

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

Quality of Work. The concept of reviewable, deliverable output at each significant development step is of great effect in promoting high quality software. We know of one project—a real time control system replacement project—which held this philosophy strongly, requiring that information be communicated between development steps only in externally reviewable forms. This project completed on schedule, suffered no backtracking, and had five bugs that lasted more than 24 hours in all of the software development. No bug lasted 48 hours.

We have observed that the work produced by a matrixed project typically has an unusual degree of cohesion, appearing to have been produced by a single mind. We believe that this occurs because the entire software development is laid out in advance in a regular framework, and because successful approaches and forms tend to be propagated from column to column.

7. Report on Experience to Date

We have used the Project Matrix approach on five separate software projects. The types of projects and rough size estimates are shown in the table below.

Project A	automated laboratory data acquisition product	25 man-years
Project B	real time control and supervisory system	60 man-years
Project C	high speed process control	15 man-years
Project D	inventory and process control	12 man-years
Project E	labor cost management database	1 man-year

Projects A and B had a long history of floundering in the extended initial phases before we became involved. These projects had been fully staffed for two and three years respectively, yet in both there was little understanding of the scope or organization of the software and no credible plan for organizing the development staff to go from the requirements/specification stage to the code. We were able to develop a Project Matrix for each project within a few weeks. The impact of the matrix on the project members was strong, immediate, and obvious.

With the introduction of the matrix, project members had clearly defined goals and systematic methods for achieving the goals, where previously both had been lacking. Additionally, the Project Matrix made it possible for each project member to understand how each piece of work related to the longer range goals of producing the whole software system. The result of this was that as pieces of work were completed, a strong sense of team spirit and high project morale developed. Most importantly, on each project, progress started to be made. This was evidenced in high quality analysis and design documents which captured the results of sound investigation, creativity, and problem solving.

In Project C, we were consulted just as the project was completing an initial requirements phase. The organization was hoping to install some modern software engineering technology with which they had little experience. The stated goal was to reduce complexity in the delivered system—a problem which had dominated previous similar work in the organization. It was our assessment that these goals were far out of reach, given the existing project management methods, schedule, and budget. To address this situation, the

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

Project Matrix was used as a vehicle to install a few carefully selected modern tools. The project is now close to completion, running somewhat ahead of schedule, and has definitively achieved its complexity reduction goals. In addition, advances in understanding systematic software development have been made.

Projects D and E are just getting underway. Project D appears to be following in the footsteps of C; we look forward confidently to its success. Project E, on the other hand, is occurring in an environment which has few of the necessary conditions for software engineering outlined in [2]. While this project may not continue to follow the plan laid out for it by the Project Matrix, we expect that an awareness of systematic methods will be retained by individuals closely associated with the project.

8. The Project Matrix as a Problem Solver

A number of very significant problems encountered in the management of software development work were characterized in a survey[4]. Of the twenty hypothesized problems listed, we have identified six for which we believe the Project Matrix provides either a complete solution or a framework upon which such a solution can be based.

Visibility. The matrix divides the work so that one can reduce questions on progress to simple done-or-not-done answers at the box level. These answers are valid if the work of each box is well-reviewed.

As a simple visibility aid, one project used a "matrix poster", a drawing on which was annotated the completion status of each box, the name of the person responsible for the box, and the projected completion date.

Project Planning. The project matrix provides a framework for collecting together detailed plans for each box. We constructed a Planning Notebook for one matrixed project, comprised of one chapter per column, and a section within the chapter for each matrix box. The box-level plans consisted of steps aimed at the production of the output objects: investigations, verification of results, drafting of document sections, reviews, and the like.

Accountability. The matrix provides an accountability structure, since the responsibility for producing the work of a box can be easily assigned to an individual or team leader. This is a proper concept of responsibility: the information flow diagram describes the information which is required as input to the task, and that information can be reviewed before responsibility is accepted.

Control. We think of project control as effective comparison of the project's work with the project plans and requirements, together with active direction to keep the work and plans consistent with one another. The visibility of the matrix work, together with the framework provided for an understandable project plan, makes the comparison aspect of this task relatively straightforward. Further control and tracking of progress can be done through the detailed plans laid out in a comprehensive Planning Notebook as described above. Finally, it is a simple matter to produce a form to be filled out weekly or monthly, asking each project member to state the percentage of his time he has spent on any box. Unexpected responses will flag potential problems which can then be investigated.

Tracing Requirements. The Subsystem Notebooks, as supported by the Information Flow diagram, show in a meaningful and non-mechanical way exactly how the requirements the project is to meet (an early row) are translated step by step to the code design and finally to the listing of the code.

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.

Planning for Maintainability. We view system maintenance as a response to modified requirements, where the modification occurs after the system has been delivered. As such, it can be seen as development displaced in time. The delivery of the Subsystem and Methodology Notebooks provide a basis for such maintenance, which can then be conducted by making modifications to the existing work in the same traceable manner as was done in the original software development.

The Project Matrix also provides some assistance in two other problems identified in the survey: Schedule Planning, as was pointed out earlier, and Cost Planning, in that the units of work that must be estimated are easy to detail. The regularity of the matrix structure provides some verification of the reasonableness of cost estimates by comparison of columns of similar nature and difficulty.

9. Conclusions

The Project Matrix is a model of the software development work of a project which can be used for project management. The matrix provides a single framework for organizing, relating, and viewing several diverse aspects of the project. It has been used successfully on several projects, serving a variety of purposes including:

- providing a framework for project planning
- identification of intermediate and final deliverables
- providing a systematic method for deriving a work breakdown structure
- providing a framework for tracking progress in terms of completed/not completed status of all activities
- supporting the tracing of requirements through all stages of software development

References

1. Comer, E. R. “Rigorous Software Engineering Standards Through Progressive Documentation”, *Proceedings of the Second Software Engineering Standards Application Workshop*, IEEE, 1983.
2. Freeman, P. and R. A. Hermon, “Lessons Learned from the Development and Application of Software Engineering Standards”, *Proceedings of the Second Software Engineering Standards Application Workshop*, IEEE, 1983.
3. DeMarco, T. *Structured Analysis and System Specification*, Yourdon Press, New York, 1978.
4. Thayer, R. H. et al. “The Challenge of Software Engineering Project Management”, *Computer*, pp 51-59, August 1980.

Originally published in *Proceedings of the Third Software Engineering Standards Application Workshop* (SESAW III), October 1984.