Center for Quality of Management Journal

REPRINT NUMBER:
RP10400

Cycle Time Reduction Special Issue

Volume 8, Number 2 **Autumn 1999**

*This page left intentionally blank
for purposes of duplex printing.*

Planning Concurrency and Managing Iteration in Projects

by Stephen Denker, Donald Steward & Tyson Browning

Reducing cycle times, managing costs and improving project management are key to competitive advantage. This paper introduces an approach known as the Dependency Structure Matrix (DSM) and discusses using the DSM to design project plans that produce greater concurrency and better iteration management. The DSM focuses management attention on the essential information transfer requirements of a project: finding critical tasks vital to begin early, developing project plans having improved throughput, removing unnecessary iteration, and simplifying project reviews, while at the same time preserving or improving deliverable quality.

Introduction

Typically we respond to the challenge of reducing project cycle time and cost by encouraging teams to *work concurrently*. Designers are now expected to work more closely with manufacturing and marketing. But concurrent project management is more than simply gathering the right team, and concurrency means more than teams taking a global view of projects and development. First of all, teams need to design how to work and interact; teams need to design the project plan itself.

Project managers have adopted techniques for planning, organizing, and monitoring the complex network of tasks in an entire project. However, the most widely used representations do not adequately describe the constraint structure underlying the project. Critical path methods (CPM) are commonly used to address these issues. But critical path analyses are inadequate to clarify an appropriate strategy. The critical path method performs a linear time/cost tradeoff for tasks on the critical (longest leadtime) path. To accelerate the project, additional resources (at greater cost) are placed on critical tasks to shorten the critical path. CPM cannot effectively handle iteration (a fundamental characteristic of product design). Further, it assumes that we can shuffle resources freely — that skilled persons are available for the asking. This is never the case.

The description we use is based on Steward's Dependency Structure Matrix (DSM).[1] We use a matrix representation (the DSM) to visualize both the sequence of relationships among the tasks to be performed and to show how they are constrained by each other. These relationships define the *structure* of a project, which is then analyzed to explore opportunities to design a workflow with streamlined intertask coordination.

The DSM concept is simply this: consider all the items of information that must be resolved to describe the final project plan and how it is to be carried out. Then consider for each item what other items it depends on. This gives us a square matrix (the DSM), as shown in Figure 1 (next page). A mark in the DSM shows where one item is needed to determine another. The DSM shows us what will be affected by any change in information content or validity.

Stephen Denker, former Senior Business Analyst in the Corporate Information Technology Group at GTE Internetworking, is a Senior Member of the IEEE and the Project Management Institute. Stephen has over thirty years industrial and business experience including both engineering and product management. He has authored more than 25 technical and business articles. Mr. Denker resigned from GTE Internetworking in November 1999 to take a self-funded early retirement and re-start his management consulting practice.

• • •

Donald Steward has degrees in physics, math and computer science, has worked in nuclear engineering, econometric modeling, and computer science, in both industry and as a university professor. His various work has always had an underlying theme of problem solving. His original work that has led to the Dependency Structure Matrix dates back to the 1960s. He has two published books, *Systems Analysis and Management: Structure, Strategy and Design*, about DSM (1981), and *Software Engineering with Systems Analysis and Design* (1987).

• • •

Tyson Browning conducts applied research and provides internal consulting on engineering process development for Lockheed Martin Tactical Aircraft Systems in Fort Worth, Texas. He previously worked with the Product Development Focus Team of the Lean Aerospace Initiate at the Massachusetts Institute of Technology. Browning holds a Ph.D. in Technology, Management and Policy from MIT.

[1] Steward, Donald, *Systems Analysis and Management: Structure, Strategy, and Design*, Petrocelli Books, 1981.

For example, constraints give structure to the patterns of linkages between customer requirements and the parameters that define a product. A DSM represents these relationships and how they are constrained by each other. Information dependencies affect the order in which tasks must be done. By analyzing information dependency structure, it is possible to develop strategies leading to shorter project cycle times and improved quality. Using this technique to visualize processes in several companies, we have developed a management strategy that focuses attention on the essential information flow requirements of a project.

Often, when we look at information interdependence, we find something quite interesting. We get blocks of items that all depend on each other, giving us no place to begin unless we start with some assumptions. All tasks within such a block are related — change any one task and all the rest will be affected. Picking any task in the block, we can follow the dependencies to any other task in that block and back again; i.e., we have an iterative circuit. This circular property is what makes it difficult for us to know what to do first. A DSM helps us decide what to assume and how to plan reviews.

An Intuitive Example — Putting on Shoes

The DSM is a spreadsheet-like square matrix with one row and column per component. Off diagonal entries indicate directed interconnections (pair dependencies). Process activities are listed in roughly chronological order so upstream activities are in the upper rows. Entries below the matrix diagonal imply information feed forwards. Above the matrix diagonal, entries represent feedbacks — the potential for iteration in a process. Thus, if rows (and corresponding columns) i and j of the DSM have no direct interconnections, entries **ij** and **ji** in the matrix will be zero or empty. If, on the other hand, both entries **ij** and **ji** are filled, this indicates two way dependency or coupling between the activities. This points to a web of "chicken and egg" problems. Many such problems exist in complex processes.

Figure 1 shows a DSM for a familiar process, putting on shoes.[2] Precedence relationships flow in a counter-clockwise direction in the matrix, so the entries below the matrix diagonal indicate, for instance, that GET SOCKS must precede PUT ON SOCKS, and GET SHOES must precede PUT ON SHOES and INSPECT SHOES. That is, reading across row 3, we see that PUT ON SOCKS depends on GET SOCKS, and reading across row 4 we see that PUT ON SHOES depends on GET SHOES and PUT ON SOCKS. Reading across row 2, we see that GET SHOES depends on INSPECT SHOES. Since this mark is above the diagonal, we make an assumption about this dependency and proceed without it. Thus, the entry above the matrix diagonal in the DSM indicates that,

[2] This non-business example about putting on shoes is used to provide an intuitive introduction to the DSM. Latter sections of this paper use examples from business.

	GET SOCKS	GET SHOES	PUT ON SOCKS	PUT ON SHOES	INSPECT SHOES
GET SOCKS	1				
GET SHOES		2			X
PUT ON SOCKS	X		3		
PUT ON SHOES		X	X	4	
INSPECT SHOES		X			5

Figure 1 shows a DSM for the process of putting on shoes.

once shoes have been inspected, they may be found wanting (e.g., scuffed up or wrong color for clothes), requiring an iteration, GET (new) SHOES.

Our goal is to resequence activities to (1) remove iterations or (2) minimize their scope. The activities GET SHOES and INSPECT SHOES are coupled through a feedback loop. However, there is no way to reorder the rows (and columns) of the DSM to get all the entries below the matrix diagonal. Failing this, we change our goal to getting any entries as close to the matrix diagonal as possible, minimizing the scope of the iteration. In Figure 1, once we GET SHOES, we go ahead and PUT ON SOCKS and PUT ON SHOES before we INSPECT SHOES. If, instead, we moved the inspection step upstream, as shown in Figure 2 below, we minimize the impact of a need to get (new) shoes. (Essentially, this act of moving activities upstream demonstrates concurrent engineering: we minimize any feedback loops in hope that we will decrease the variance in total process cycle time.) While in this example we could not eliminate iteration entirely, often it is possible to reduce the number of potential iterations substantially by resequencing rows and columns.

	GET SHOES	INSPECT SHOES	GET SOCKS	PUT ON SOCKS	PUT ON SHOES
GET SHOES	1	X			
INSPECT SHOES	X	2			
GET SOCKS			3		
PUT ON SOCKS			X	4	
PUT ON SHOES	X			X	5

Figure 2 shows a DSM representing a restructuring of the process of putting on shoes.

The DSM also indicates which activities can be accomplished in parallel without causing additional iteration. For example, in the diagrams above, GET SOCKS and GET SHOES can be done simultaneously, as can INSPECT SHOES and PUT ON SOCKS (if we have enough resources!). Doing things concurrently is often seen as a way to reduce cycle time. But if we choose activities to work in parallel without first considering their dependencies, this can lead to additional iteration and increased cycle time. Arbitrarily doing tasks concurrently does not guarantee reduced cycle time.

Activities capable of execution in parallel from an information flow perspective may be held up by resource constraints. A more complete analysis must account for these constraints. For example, the DSM above indicates that GET SOCKS and GET SHOES can occur in parallel. If we only have one person available for both activities, we may not be able to do both at once.

Off-diagonal entries do not have to be binary (i.e., present or not); they can be numbers from 0 to 1, from 0 to 9, etc. These numbers convey information regarding specific pair dependencies, such as probability of iteration, percent rework, amount of data flow, type of data flow, sensitivity of downstream activity to change in input, etc.

Activities need not be limited to finish-to-start type relationships; i.e., activities can be partially overlapped in certain cases, although these decisions should be based on the nature of information production and use by the upstream and downstream activities, respectively. Assigning durations to the activities in the DSM, along with knowledge of serial, parallel, and coupled activities, can enable critical path calculations.

Concurrency has Structure

All work has structure. Every process as realized has structure. Structure constrains a process in many ways. Wrong structures limit and restrain. Successful businesses need competitive structures. We transform information dependency structure into a project plan used to coordinate team tasks.

By visualizing the dependency structure required by information flow, we can make decisions more wisely. Not keeping good track of information dependencies, and whether they are being satisfied, causes unnecessary project risks, overruns, and failure to meet requirements. With DSMs, assumptions are made explicit. A DSM clearly shows which information items are dependent on assumptions and where they should be verified. By creating a more detailed description we explicitly map all aspects of a problem. To be most useful, a project plan representation must include not only the sequence of tasks but also task dependencies and potential iterations.

A DSM serves as the template for ordering tasks and managing information dependencies. Analyzing information dependency structure can provide a much more efficient and predictable process.

Sharing a DSM makes the situation visible. At any time, everyone can see what has been done and what is ready to be done. The DSM also shows what assumptions have been made and where they are to be reviewed. We can also see what depends on assumptions that have not yet been verified. Project inefficiencies occur when the structure of the information flow inherent in the project clashes with the information flow enabled by the project organization. Cycle time is often wasted by holdups for critical information missing because no prior analysis was made of the information dependencies inherent in the process. If the situation changes, a DSM shows what must change as a consequence and who must be informed. The picture provided by the DSM empowers project participants to provide information flow required by the project. Then, information can move freely and effectively. Everyone can see the effect of their own contribution and how it fits into the whole, giving a greater sense of ownership and pride in their work. If people are to be truly empowered, they need to have situation visibility. Effective exchange of information is key to coordinating collaborative development through concurrency. Accomplishing this exchange is fundamental.

So how do we design project plans that we can schedule? By using the DSM first, we make the process structure explicit and understandable. The basic principle is to arrange tasks around the exchange of information. Ask this simple question to make the structure explicit: "What information do I need to do my job?" In this way we can build a DSM — matrix row by matrix row. Then, the DSM columns answer the harder question: "What information do I owe others so they can do their job?" These questions answered, we can create organizations structured for better information exchange, better division of responsibility, and greater concurrency. We can better understand and know what to communicate, when and to whom. We can see clearly our interfaces with other groups to receive the information we need when we each need it.

Iteration in Projects

Design relationships between parameters describe a product and how it is made. Together, these relationships form a *descriptive* statement of the product constraints. Typically an information exchange graph describing constraints between design parameters will contain dependency circuits because these parameters live simultaneously in the final product. Because they must be simultaneously satisfied in the product design, they likely contain iterative loops, or circuits. Wherever information flow involves dependency circuits, there is no place we can start without having parameters that depend upon some other parameter. To design a product we undertake a set of tasks that will set values on these parameters. The project to do this is a *prescriptive* process. We carry out tasks over time. Tasks must be in sequential or in parallel order.

The DSM method derives a prescription for tasks from the description of constraints. Wherever information exchange graphs involve dependency circuits, there is no place we can start without having parameters that do not depend upon some other parameter. Most parameters within dependency circuits must be determined iteratively. This iteration is not unique. To begin an iterative process, we must decide where assumptions will be used so dependency circuits would be broken. This allows some tasks to begin by using assumptions rather than by waiting for the completion of the (successor) task that would have supplied the information. We say "dependency circuits are torn" by assumptions and then unwrapped into a finite sequence of iterations that can be carried out over time. Without task circuits, project plans can be scheduled. We tear where our judgment suggests we can both use assumptions while most effectively breaking up dependency circuit structures.

In the next section of this paper, we describe our technique of transforming descriptive maps (constraints) into prescriptive maps (project schedules).

Example — Managing Iteration by Unwrapping Circuits

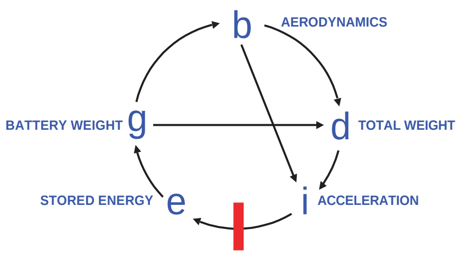


Figure 3 describes constraints between design parameters b, d, i, e & g.

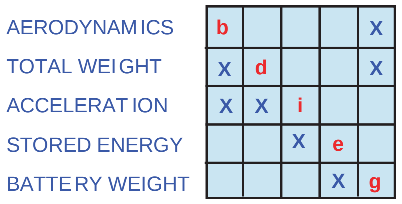
Figure 3 is a graph relating the parameters required to design an electric car.[3] For instance, the graph in Figure 3 indicates the following:

- the battery weight parameter **g** depends on the stored energy parameter **e**;
- the aerodynamics parameter **b** depends on the battery weight parameter **g**;
- the total weight parameter **d** depends on the battery weight parameter **g** and the aerodynamics parameter **b**;
- the acceleration parameter **i** depends on the aerodynamics parameter **b** and the total weight parameter **d**; and
- the stored energy parameter **e** depends on the acceleration parameter **i**.[4]

Given the circularities obvious in the Figure 3 graph, where should one start the process of choosing the design parameters? The first step is to create a matrix representing the same information shown in the graph. A matrix and a graph are alternative descriptions of the same situation; thus, the graph of Figure 3 can be represented as a descriptive DSM as shown in Figure 4.

[3] Ignore the vertical bar between **i** and **e** for a moment.

[4] Some readers may feel more comfortable with another example. You can make up your own example for the graph in Figure 3. For example, the same graph might also represent the process of designing a new system, if we assigned the following names to the nodes of the graph: e = user requirements; g = functional spec; b = hardware design; d= software design; and i = performance spec.



AERODYNAMICS	b				X
TOTAL WEIGHT	X	d			X
ACCELERATION	X	X	i		
STORED ENERGY			X	e	
BATTERY WEIGHT				X	g

Figure 4 is a matrix showing constraints between design parameters.

(Remember, as described in the previous section, a DSM is a square matrix with one row and column per component. Since DSM rows and columns are arranged in exactly the same order, we identify them both by the letter in the diagonal boxes of the matrix. Off diagonal entries indicate directed dependencies. Thus, if the component represented by row **i** is dependent on the component in column **j**, the entry in row **i**, column **j** will have an entry.)

Having created the matrix of Figure 4, we can rethink the process and manipulate the matrix to represent a situation where we estimate the parameter for stored energy (call this estimate $\mathbf{e_1}$) when we don't know a value for the acceleration parameter (**i**). Such a restructuring of the situation is shown in Figure 5.

STORED ENERGY	E				E_1
BATTERY WEIGHT	X	G			X
AERODYNAMICS		X	B		
TOTAL WEIGHT		X	X	D	
ACCELERATION			X	X	I

Figure 5 is a prescriptive matrix showing relationships between design tasks.

$\mathbf{E_1}$ represents the design task **E** that uses an estimate of **e** ($\mathbf{e_1}$) made without having a value for **i**. $\mathbf{e_1}$ is the first pass estimated value for design parameter **e**. This is equivalent to cutting (or tearing) the graph in Figure 3 at the point of the vertical bar between **e** an **i**.[5] Thus, we derive the prescriptive design task matrix shown in Figure 5. **E**, **G**, **B**, **D** and **I** represent design tasks required to determine parameters **e**, **g** , **b**, **d**, and **i**. Figure 5 prescribes the relationships between the design task sequence ($\mathbf{E_1(e_1)}$, **G**, **B**, **D**, and **I**) that determines parameters **b**, **d**, **i**, **e** and **g**) described by the matrix in Figure 4.

[5] How you decide to tear the graph or matrix between **i** and **e** is an issue beyond the scope of this paper. (This issue is addressed by Steward.[1])

After tearing we can unwrap dependency circuits into a linear sequence of task iterations as shown in Figure 6. The resulting prescriptive project plan does not contain circuits. We can use it to develop a critical path schedule for the tasks. We also know exactly what and where we used estimates to accomplish this.

$$\underbrace{E_1(e_1) > G_1 > B_1 > D_1 > I_1}_{\text{Iteration 1}} > \text{REVIEW}_1 > \underbrace{E_2 > G_2 > B_2 > D_2 > I_2}_{\text{Iteration 2}} > \text{REVIEW}_2$$

Figure 6 is one prescriptive project plan for design tasks (unwrapped circuits).

At the end of each iteration we have tentatively inserted reviews to determine whether the estimate $\mathbf{e_1}$ for parameter **e** was adequate or whether another iteration is required. We began with an estimate $\mathbf{e_1}$. Our first review assesses whether this estimate was satisfactory. If not, we make another iteration ($\mathbf{E_2} \to \mathbf{G_2} \to \mathbf{B_2} \to \mathbf{D_2} \to \mathbf{I_2}$). If we need more or less iterations than originally planned, the prescriptive DSM will show all the necessary predecessors.

Suppose we also decide to estimate *Aerodynamics* ($\mathbf{b_1}$). $\mathbf{e_1}$ and $\mathbf{b_1}$ are the first pass estimated values for design parameters **e** and **b**. We use estimates $\mathbf{b_1}$ and $\mathbf{e_1}$ for tasks $\mathbf{B_1(b_1)}$ and $\mathbf{E_1(e_1)}$ shown in Figure 7.

STORED ENERGY	E				E_1
BATTERY WEIGHT	X	G			X
AERODYNAMICS		B_1	B		
TOTAL WEIGHT		X	X	D	
ACCELERATION			X	X	I

Figure 7 is another prescriptive matrix showing relationships between design tasks.

Now an additional tear in the arc between parameters **g** and **b** appears below the matrix diagonal. We have made this tear to show one way to gain more parallelism. This gives the unwrapped prescriptive project plan for design tasks shown in Figure 8.

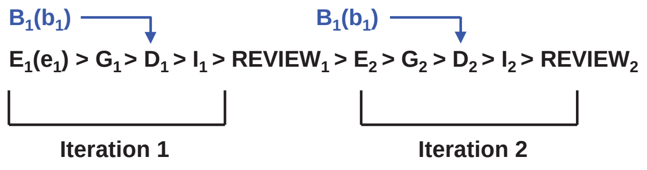

$B_1(b_1)$
$B_1(b_1)$
$E_1(e_1) > G_1 > D_1 > I_1 > REVIEW_1 > E_2 > G_2 > D_2 > I_2 > REVIEW_2$
Iteration 1
Iteration 2

Figure 8 is another prescriptive project plan for design tasks.

Alternatively, suppose after the first review we find that the value for parameter **b** must be revised from its original estimated value ($\mathbf{b}_1$), we could revert to the sequence shown in Figure 9.

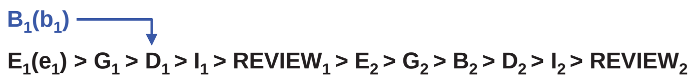

$B_1(b_1)$
$E_1(e_1) > G_1 > D_1 > I_1 > REVIEW_1 > E_2 > G_2 > B_2 > D_2 > I_2 > REVIEW_2$

Figure 9 is yet another prescriptive project plan for design tasks.

Example — Increasing Concurrency

When looking at a project plan to see how it might be transformed into a concurrent process, it is possible to fall into a trap. Commonly we group tasks into major phases, with a verification review task at the end of each phase. Project phases are often generic (e.g. preliminary, intermediate, and final) and not tailored to the structure of a particular problem. Verification becomes the gateway to entering the next phase. A DSM can show us how to bypass this lock — how to reduce the number of reviews or remove them altogether.

We begin by removing all review tasks. Instead of showing any group of tasks as being dependent upon a verification review task that ends a previous phase, we show precisely which tasks in earlier phases these later tasks directly depend upon. Only after analysis, do we add verification tasks. Reviews are tailored to the structure of information dependencies rather than to a generic review process. It now becomes possible to have fewer, simpler reviews of smaller project phases and significantly increase concurrency. Breaking large generic review cycles into problem-specific, micro reviews can in itself result in significant cycle time reduction.

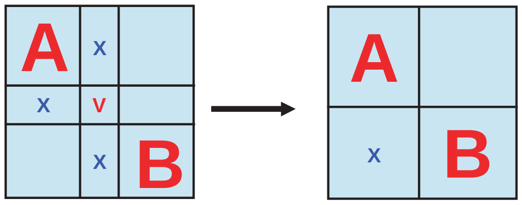


Figure 10 is a set of partitioned matrices showing verification and its removal.

In the DSM, blocks represent circuits (closed loops) in the graph. For example, in the left DSM in Figure 10, **V** is a verification task for the a group of tasks, **A**. Block **B** is dependent only upon **V** and **V** depends only on block **A**. (The **X**s represent these dependencies.) In the right DSM we have eliminated **V** and have shown the direct dependency of block **B** on block **A**.

Now we can see where concurrency is appropriate. To further illustrate, consider the set of DSMs shown in Figure 11. After partitioning we may find a structure similar to the left most DSM.

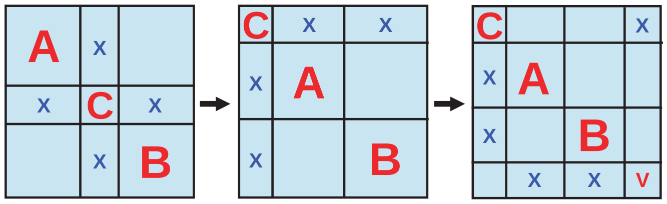


Figure 11 is a set of partitioned matrices suggesting better placement of verification.

Consider just the information dependencies that couple blocks **A** and **B**. Blocks **A** and **B** are coupled by block **C**. This DSM could be simply reordered to obtain the DSM in the middle of Figure 11. Block **C** sets the requirements for **A** and **B**. Once the tasks in block **C** are carried out, the information requirements for **A** and **B** are known, and tasks in blocks **A** and **B** can be carried out concurrently. But blocks **A** and **B** are coupled by feedback through block **C**. To deal with this, we insert a final verification task **V** replacing any feedback dependence on block **C**, as shown by the right-most DSM. This strategy is explained at great length in by Baldwin and Clark.[6]

[6] Baldwin, Carliss Y. and Clark, Kim B., *Design Rules: The Power of Modularity*, Cambridge, MIT Press, 1999.

Summary

We have introduced the concept of information dependency structure. Once this structure is made clear using the DSM, we can construct project plans that enable us to perform tasks and share information more effectively and at higher levels of quality.

Bibliography

Austin S., Baldwin A., Li B. and Waskett P. A paper accepted by *Project Management Journal* — "Application of the Analytical Design Planning Technique to Construction Project Management."

Eppinger, S.D., Whitney, D. E., Smith, R. P. and Gebala, D.A., "A Model-Based Method for Organising Tasks in Product Development," *Research in Engineering Design*, Vol. 6, No. 1 (1994), pp 1-13.

Rogers, J. L. and Padula, S. L., *An Intelligent Advisor for the Design Manager: Technical Memorandum 101558*, NASA, 1989.

The Center for Quality of Management Journal is a forum for disseminating the experience of organizations learning to implement modern management practices. It seeks to capture experiences and ideas that may be useful to others working to create customer-driven, continuously improving organizations.

The CQM Journal is refereed. However, it is not an academic publication. Experiences and ideas will be published if they seem likely to be useful to others seeking to improve their organizations.

Send to:

The Center for Quality of Management Journal
Editorial Department
One Alewife Center, Suite 450
Cambridge, MA 02140
Tel. 617-873-8950 Fax 617-873-8980
E-mail: publications@cqm.org

If you have thoughts for a paper and you would like to discuss it with us, please write, call or submit an outline. We welcome your ideas.

Final Manuscript Requirements:

Entire manuscript should be double-spaced, including footnotes, references, etc. Text should include all the elements listed below. Generally, The CQM Journal follows the editorial principles of The Chicago Manual of Style. We strongly prefer submissions in eletronic format for all text and as many of the figures as possible. IBM-based software (particularly Microsoft Word for Windows) is preferable to Macintosh-based software if you have a choice, but is by no means a requirement.

Please include:

1. Title page, stating the type of article (e.g., 7-Step case study, research paper, short communication, letter to the editor, etc.), main title, subtitle, and authors' full name(s), affiliation(s), and the address/phone/fax of the submitting author;

2. All needed figures, tables, and photographs (see below);

3. Footnotes (if appropriate), numbered consecutively from the beginning to the end of the article;

4. Reference list, if appropriate.

Figures, Tables and Photographs:

If you can, insert each figure or table into the text where you would like it to fall. Figures should be composed to conform to one of two widths: 3 1/8 or 6 1/2 inches. The maximum height for any figure is 9 3/8 inches. Text within figures should not be smaller than 5 points and lines not less than 1/4 point at the figure's final size. Figures should labeled with the figure number underneath and title on top. Be sure that the text mentions each figure or table.

Please retain separate PICT or TIFF files of figures generated in drawing programs and a file with the text only for final submission.

CENTER FOR QUALITY of MANAGEMENT JOURNAL