Document-Driven Management of Knowledge and Technology Transfer: Denmark's CIM/GEMS Project in Computer-Integrated Manufacturing—Part 2

Ellen McDaniel, Robert E. Young, and Johan Vesterager

Abstract—This article is the second of two articles (see E. McDaniel et al., IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION, June 1991, pp. 83–93) discussing ways to create system documentation and use it to drive and manage technical development in technology-transfer projects. This article again explores the two-year Danish CIM/GEMS project in computer-integrated manufacturing (CIM) at the Technical University of Denmark to show how writing is an act of technology and knowledge representation. A vehicle of their transfer to a user community, and, if successful, an accommodation of technology to its users. The authors address creators and users of system documentation who need documentation for CIM implementation. The authors argue that documentation is often better for representing and explaining a CIM system than the actual system itself, and they recommend that documentation production not be viewed as a separate, end-of-project activity but as an integrated part of technical development. Planned and regular documentation production can in fact be a stimulus and aid to technical development, possibly even shortening the project life cycle.

INTRODUCTION

No one lately has discussed the problematic role of the writing researcher more astutely than Carl Herndl in his article “Writing Ethnography: Representation, Rhetoric, and Institutional Practices,” which appears in the March 1991 issue of College English. As researchers, we are governed and conditioned by accepted institutional practices, which essentially are textual practices, by which we must turn experience into textual representation, or events into accounts. The “participant observer” role of the sociologist, the anthropologist, and the ethnographer is the role we must assume as well, but it is a role that establishes authority by contrary means [1]. The researcher must claim both the subjective credibility of the participant (“I was there”) as well as the objective authority of the observer (“I was not really there”). Such researchers necessarily, but perhaps erroneously, assert that their representations are disinterested assessments of experiences, informed yet unblurred by their participation in them. Perhaps such studies should carry warnings to the reader of representation’s possible limits and distortions.

Research in technical writing—and writing in general—is doubly troubled because writing is already a representation of things, methods, and technology, so research in technical writing is really a representation of a representation, an account of an account. The case study we describe in this article is one such attempt to discuss in writing how writers write to represent and explain a technology. We do not doubt that there is still value to be gained from research of this kind, even if the reporting is twice removed from the technical events that give rise to it. Still, it bears hearing James Clifford again, whom Herndl also quotes:

How is unruly experience transformed into an authoritative written account? How, precisely, is a garrulous overdetermined, cross-cultural encounter shot through with power relations and personal cross purposes circumscribed as an adequate version of a more-or-less discrete “other world” composed by an individual author [2]?

To write about technology and then to write about the writing erects a great deal of institutional practice and research convention between the empirical reality of the technical project itself and the article discussing how the project was documented. And yet, it is the responsibility of writing researchers to explain as precisely as we can just how experience is transformed into an authoritative account. In other words, what is successful writing and how is it done?

Herndl also demonstrates how cultural and historical practices within institutions influence our manner of telling. His exposé of the substantial institutional practice in Stephen Doheny-Farina’s article (“Writing in an Emerging Organization: An Ethnographic Study”) makes it clear that, at the very least, researchers should tell their readers what research practices shape the representation or “version” of events they are reading (though Herndl shows how clearly they show through anyway). Still, I would argue—and this is not contrary to Herndl’s observations—that the written account is the revelation and expression of technical events, so much so that it is part of the technology itself and to many readers who never witness the technology firsthand, it is the technology. Certainly, writing is a practice—convention-bound activity, and it proceeds, as Edward Said says and Herndl quotes, “according to a detailed logic governed not simply by empirical reality but by a battery of desires, repressions, investments, and projections” [3]. Nevertheless, the written

Manuscript received September 1991.
E. McDaniel is with North Carolina State University, College of Engineering, Engineering Computer Operations, Raleigh, NC 27695-7901.
R. E. Young is with the Department of Industrial Engineering, North Carolina State University, Raleigh, NC 27695-7906.
J. Vesterager is with the Institute of Production Management and Industrial Engineering, The Technical University of Denmark, DK-2800 Lyngby, Denmark.
IEEE Log Number 9200129.
account is an orderly text and progression of events articulating what the developers say the technology is. The writing is less garrulous than the development and has a single purpose sculpted from the cross purposes of experience. In this article, we discuss ways to achieve this expressive consensus and restriction of scope in technical development, arriving at an accurate representation of the technology as possible. We also assert that by constructing the telling view and written account of the technology, the writer is more an architect than an historian of technological development.

ACCOMMODATING TECHNOLOGY TO THE USER

David Dobrin has given the technical writing profession its most workable and accurate definition of technical writing, which is “writing that accommodates technology to the user.” He further refines this definition in his book Writing and Technique and argues that

... an examination of technical writing should, in effect, be a natural history of technical writing ... The examination would begin where someone conceives the need to accommodate, adorning the relationships of power and perception that caused this conception. The examination would end where the accommodation is completed. Along the way, the examination would follow the traces of the accommodations left in human relationships. This kind of research is very difficult, for penetrating groups of which you are not a member requires learning a new way of thinking ... The way they handle technical writing at Kodak is very different from the way they do it at Corning, and each way is tied up with the corporation’s organization, its self-image, its valuations of judgment and so on. But this research would address itself more directly to an understanding of technical writing in its quiddity [4].

This two-part article tries to follow Dobrin’s prescription. Part 1 of this paper, which appeared in the June 1991 issue of the IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION, looked at the function of technical writing within a specific organization, the Danish CIM/GEMS project (General Methods for Specific Solutions in Computer-Integrated Manufacturing) conducted at the Technical University of Denmark in 1986–1989 [5]. The writing that was done for the CIM/GEMS project will not accommodate any technology other than the one developed by that project. As such, it is one of a kind, a unique product or “specimen.” However, in the “natural history” of technical writing, the writing done for that project is like the writing done in similar CIM-development projects. The writing is an act of technology and knowledge representation, a vehicle of their transfer to a user community, and, if successful, an accommodation of the technology to its users.

This second part continues the discussion that Part 1 began, that is, to generalize from the particular writing instance of that project a writing strategy and methodology that would succeed as a general solution technology in similar development efforts. There is a technological sameness among CIM systems, like all classes of technical systems, but the individual system is also always unique and can be differentiated from other members of its class by how it is configured to manufacture a unique product. All such systems are composed of both standard and custom elements, and so is the documentation that supports and represents them. Writers must be aided by general strategies, approaches, methods, and tools, even if their product is customized documentation. Technical writing is a kind of engineering, and writers succeed by applying appropriate solution technology within the problem domain. Because each problem is different—the quiddity of technological development—accommodating technology to users is improved through practice. Skill develops through the repeated drill of using writing to solve or communicate solutions to problems. With this skill and experience as a foundation for forming and structuring solutions, writers can then penetrate the groups they would write for by learning the “new way of thinking” required, that is, the nature of the particular technology to be written about and the specific user audience that would receive it.

Henrietta Nickels Shirk reminds us in her article, “Technical Writing’s Roots in Computer Science” that most technical documentation is destined for disposal. This is the price of documentation’s uniqueness and customization. Its life, usefulness, and value are finite, and once these are exhausted, it does not bear keeping or reading except as an archive or a starting place (a model or blueprint) from which to expand or develop. Technical writing lives in the present and has value because it is current. “Old” technical writing gives us the history of technology and of the craft of technical writing, but it has little value as an artifact [6]. Technical writing research helps writers learn to make better records, blueprints, representations, and accommodations of technology by learning also how to accept and work within the constraints of technological development, its currency (in time, cost, and resources), and its development imperatives. In addition to producing better written documentation, research in technical writing should help us learn to write value-added documentation, that is, writing that improves productivity and adds value to the process and products of technology development and transfer.

SYSTEM AND USER DOCUMENTATION

The term technical documentation refers to textual and graphical information, recorded on paper or electronic medium, that represents and explains technical products or systems. Technical documentation usually includes both system and user documentation. System documentation generally refers to materials that are written and used during the making of the product before it is turned over to the user. This documentation is the product or system’s pre-physical characterization, its existence or representation as a “paper” system before and during its construction. System documentation also describes and explains the technical activities that produced the product or system, showing what it is, how it was made, and the rationale for decisions made during its construction. This documentation is a record of the complete technical endeavor that produced the finished and working product or system.
User documentation refers to the materials that are written to follow this endeavor and to support the activities of the receivers or purchasers of the product or system, the end user. User documentation describes the function of the product or system to these users and instructs them in how to employ or implement it effectively. This documentation makes the product or system both operational and productive.

The research literature in technical communication has had much to say about user documentation, particularly the manual, mainly because that is what technical writers spend most of their time writing. Technical writers are usually hired to sit at the end of a technical production line and await some technological output for which they will write a user’s manual. In better writing situations, technical writers are brought into technical development earlier and allowed to see how the product or system was made and why (which may permit them to contribute to the writing of system documentation, though often not enough). Occasionally, they may even participate in product or system testing. Nevertheless, it is still most common for technical writers to work apart from technical developers and users, which leaves them unable to learn the “new way of thinking” they need to write better documentation. As a result, they are unable to contribute to the construction of knowledge or technology during development, or to participate significantly in the activities that transfer knowledge and technology to the user. This is unfortunate, for writing is a critical tool and method for constructing the technology itself, not just for producing in its aftermath some manual (usually poor) for using it.

David Dobrin’s definition of technical writing emphasizes the user of documentation, who is a doer and a learner, not just a reader. “Using” a document requires reading, thinking about, and applying knowledge that was constructed and formulated by developers in the writing or preparation of the document. The word “user,” Dobrin says, “reflects the fact that technical writing exists within a system that measures actions, people, and things by the criterion of use and that technical writing is part of the technology itself and is, in essence, the experience of the technology” [4]. However, there are many users along the way in the making of a technical product or system, in particular, the makers themselves, who produce and use the documentation that we call “system documentation.” Other users of system documentation are those people who install, extend, and maintain the finished system; those who use the system as a basis or model for new system design; and those who consult system documentation for educational (knowledge) purposes to understand specific engineering principles, methods, technology, etc. In short, a user is by no means always an “end” user. System documentation has many uses and purposes, but it has remained a “no man’s land” to technical writers.

Causes and Results of Poor Documentation

There is a sense that technical documentation—both system and user documentation—is poor. Poor documentation figures prominently in the list of common problems that the U.S. Air Force has identified as interfering with the production and maintenance of computer-integrated manufacturing systems. The list appears as follows, and the items on it will be recognized as problems plaguing technical development in general and not just that of CIM development.

- Limited management visibility.
- Acquisition cost and schedule overruns.
- Inability to meet performance specifications.
- Lack of responsiveness to user requirements.
- Obscure programming.
- Inadequate testing.
- Inadequate transition support.

**Inadequate or nonexistent documentation** [authors’ emphasis] [7].

There are many reasons why documentation is not produced or is inadequate when it is. First, because it is not planned as part of the system and is usually done last, the writing holds no interest for developers, and there is often poor recall of the work that was done. Often, technical developers are not available to write up their work because they have moved on to new assignments or have left the project or organization altogether. Second, people do not like to write, do not write well, or have too little time to write. Strong technical writing skill is not commonplace. It is also a time-consuming and labor-intensive activity to identify and write adequately for a user audience. Third, documentation is not a managed or organized activity that has people working collectively to create effective communication vehicles for their purposes. Organizational writing is often a piecemeal and separatist activity with no governing vision or supervision that ensures that written products do what they are intended to do. Fourth, developers usually do not apply the same effort, tools, technology, and resources to document the work as they perform it. Often, management does not emphasize or support documentation with adequate resources and communication technology, which can speed the endeavor and improve the quality of its products.

Finally, and most important, there is a common misconception that technical development produces only the technical thing itself and that documentation is not really a part of technical creation nor is it important. The truth is that what is valuable about the thing produced is the knowledge to understand and use it effectively. Technology is this knowledge—<em>technē</em> and <em>logia</em> together—and written communication still expresses and transfers this technology more widely and effectively than any other means. Ironically, most of the deliverables produced by technical research and development projects are documents; yet, they generally are the weakest feature of the system or product produced, showing the least thought and poorest workmanship.

**Need for Improved Documentation**

This past inattention to documentation is beginning to disappear for several, basically financial, reasons. Technical management is becoming aware that remediating the last problem on the U.S. Air Force’s list above could possibly solve some of the other problems listed as well. A large part of a technical system’s cost is in its maintenance. Mainte-
nance—which is not simply repair but support, extension, change, or replication of the system—is impossible without clear system documentation showing how the system is put together. Other reasons for supporting documentation include the following:

- Much about complex and sophisticated technical systems is "unseeable" or intangible (e.g., miniaturized parts, programming, etc.), and documentation is needed to represent or explain them.
- No one person can know all there is to know about complex technical systems, and no one person ever builds one of them. They are created by many people performing many tasks over a long period of time. These people obviously do not stay with the product or system when it is turned over to the user, so what they know about it must accompany the system in some way, and the best way we know of is through documentation.
- A large time and resource investment is made in the research, development, testing, and implementation of technical products and systems, and it is imperative not to waste this investment by doing any of these over again. Documentation helps ensure that people do not "reinvent the wheel."
- Overall costs are less for well-documented systems and their reliability higher. More errors are discovered and corrected if sound documentation accompanies periodic technical reviews during system development, thus making these systems more reliable and cost-effective [8].

Documentation is actually a form of logical modeling: thus, the act of writing a document is an act of modeling. Writing is also a form of technical reviewing and often locates errors and inconsistencies in the design and/or configuration logic of the system.

**DOCUMENTING THE CIM/GEMS PROJECT**

On the Danish CIM/GEMS project, we had three vehicles for knowledge and technology transfer, all of which had particular strengths and weaknesses. The sample laboratory factory we built at the university was useful as a physical display of the completed technology and was the best way to show that the CIM technology we developed worked. However, this physical system could not reveal how it was constructed or why it was constructed the way it was. Also, it was not transportable. The second means—short courses and seminars for industry participants—was the best means for exchanging knowledge on the subject of CIM and of collectively engaging in idea formulation, problem analysis, and solution strategy. However, such discussions were usually general in nature, and the courses themselves were difficult to arrange and schedule and were confined to one site. The shortcoming of documentation—our third means—was its limitation as a paper system. The success of the technology was proved in the finished and working physical system. Nevertheless, documentation permitted the participating companies to repeat this success by giving them a blueprint to follow in designing their own factories. The documentation we produced captured critical design considerations and expertise. It explained the unseeable, yielded easily and inexpensively to study, and was transportable.

The CIM system we developed was a single functioning technology, a custom system created only once. However, as our name implied (GEMS—General Methods for Specific Solutions), we were interested in the multiple adaptation of this CIM technology in the workplace. The six companies in Table 1 were among the industry participants who actually committed to and completed CIM projects, which were run in parallel with our own using the technology we developed. Many other companies participated in the industry coalition that followed the work of the two-year project, attending our short courses and receiving our documentation.

The purpose of the technical documentation was to make sure that the singular technology we created could be copied, imitated, extended, changed, refined, enlarged, and, of course, used once it was set in place. It had to be thorough and effective enough for people in another place at another time to use in creating something similar or equivalent, but not the same. The following discussion gives examples of ways that we did this and also explains how we used documents to avoid or solve some of the problems on the U.S. Air Force's list previously mentioned. There is no way to represent all that was done, much of which was the ordinary, day-to-day operation of collecting, distilling, and recording information and writing it into readable form. However, some of the more valuable strategies and procedures we followed are discussed here as a means for adding value to documentation and streamlining technical development.

**Merging Project and Technical Management**

Part 1 of this article made the case for viewing document creation as a technical task and for permitting it to drive and manage technical development. For most of the phases of a project's life cycle, the system under construction is not a physical or tangible thing. Rather, it exists on paper, in lists of requirements and specifications, in written and graphical descriptions, and in flow diagrams and models. In essence, these paper resources are both the system itself and the knowledge used to build it. Although the CIM/GEMS project specified documentation as one of its five deliverables, the other four (the ongoing workshop of synchronized development projects, the short courses and seminars, the system prototypes and user interfaces, and the final working factory) all require documentation to make them effective vehicles of knowledge and technology transfer. Thus, the project organized itself around the delivery of this documentation, which stood as a transportable representation of the system and was often better at showing what the system was than the system itself.

Most development projects have two kinds of activities they undertake: project management tasks and technical tasks. Although system documentation usually refers to the documentation of the latter tasks, it is useful to include project management documentation in system documentation. A common problem in development projects is that management and technical efforts often diverge and become two separate projects, a right and left hand not knowing what the other
TABLE 1  
CIM SYSTEMS BUILT BY COMPANIES IN THE CIM/GEMS INDUSTRIAL COALITION

<table>
<thead>
<tr>
<th>COMPANY</th>
<th>PRODUCT</th>
<th>NUMBER OF EMPLOYEES</th>
<th>CIM PROJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfa-Laval Separation</td>
<td>Decanter-separators</td>
<td>200</td>
<td>1) Tool Mgt. System Prototype</td>
</tr>
<tr>
<td>Brodrene Gram</td>
<td>Refriger. Equip. &amp; Plants</td>
<td>1730</td>
<td>2) Planning Information System for Methods Engineering</td>
</tr>
<tr>
<td>Parup Electronics</td>
<td>Imaging Systems for Printing Industry</td>
<td>175</td>
<td>1) Order Configuration &amp; Entry System</td>
</tr>
<tr>
<td>Solar Armatur-produktion</td>
<td>Lighting Equipment &amp; Lamps</td>
<td>230</td>
<td>2) Manufacturing Material Management System</td>
</tr>
<tr>
<td>Burmeister &amp; Wain Shipyard</td>
<td>Bulkcarriers &amp; Tankers</td>
<td>1600</td>
<td>Order Entry &amp; Shopfloor Control System</td>
</tr>
<tr>
<td>Consolidated Diesel</td>
<td>Diesel Engines</td>
<td>1000</td>
<td>Shopfloor Control System</td>
</tr>
</tbody>
</table>

is doing (see again the U.S. Air Force’s list previously mentioned). System documentation that keeps management and technical activities synchronized and their personnel apprised of the work of the other can aid the progress of technical development.

**Project Start-Up and Planning**

One particularly troublesome period for management, technical, and writing stuff alike is the project’s beginning. In fact, it is here that projects generally make or break themselves. Writers most often feel superfluous in the beginning of a project because they believe that writing is premature at this point and that their job is to listen and learn. They too often assume that the developers are clear in what they are doing, and the writer’s job is to figure out what they already know. If a proposal or recommendation preceded the development project, the writer may be even more likely to assume that the work is completely planned and laid out. Developers may believe this too. However, once actual parameters are set for the project—budgets, schedule, staff, equipment, etc.—in essence, when the project is in its negotiated state for work, the project may take on a much-changed configuration, and the plan must be re-thought. Often, developers look back at their proposal to find direction for this difficult phase only to discover that there is little there to help them. The proposal is speculative and conjectural whereas the funded project is incontrovertible. Plans that are conjectural are very different from plans that actually must be followed.

Start-up is very slow in projects for these reasons, and much time is spent in this phase. This phase might be called the “Prufrock Phase” with its hundred indecisions, visions and revisions, and the major question is “How should I begin?” A writer’s job is critical at the beginning of a project, not for documenting the system but for documenting the project itself. If the project begins in uncertainty, then the writer’s job is to lead project members out of it. A writer is educated in gathering information, analyzing it, defining purposes, organizing ideas, and composing coherent and ordered definitions in text. Those skills can be brought to bear in helping a project define itself to itself, to management outside the project, and to any audience that may be witnesses or beneficiaries of the work.

Thus, if the project needs an identity, a scope, and a strategy, the writer must try to discern from the project members what these should be and put them on paper. The writer’s first responsibility is to see the project whole and to create the telling view of it. The writer gathers what she needs to begin conceiving these documents through journalistic methods—relying on short interviews with developers and notes from technical meetings (also taping those that are particularly technical to play back when writing), some directed reading and study in the technical area, and informal discussions in offices and electronic mail. These journalistic techniques are noninvasive means of information-gathering that take very little time away from the developers. They can help the writer glean the visions and revisions that people have about the project and arrive at a plan to put on the table.

An organization chart and a project summary were the two central documents we produced in the project’s start-up phase. The chart appears in Part 1 of this article and is reproduced here in Fig. 1. The project summary included most of the information found in the background of the CIM/GEMS Project section, also in Part 1. The organization chart—usually a management document—is not a typical organization chart of seniority levels in a company but is a technical document in itself.1 The chart was the product of information gathered from early project meetings, which, incidentally, grew increasingly frustrated and argumentative as project members grappled with what the project was and would do. Because the network of university and industrial personnel was complex—composed as it was of project developers, company participants, and parallel projects—we had to have for ourselves and to present to the participating companies a conceptual view of the project at large.

The view we finally agreed on is presented in the chart, which diagrams the parallel nature of the development projects, that is, the “internal” university CIM project and the several “external” CIM projects run by companies in our industrial coalition. We also 1) identify the project deliverables, 2)
indicate that the knowledge and technology transfer was a two-way exchange, and 3) structure a way for the companies to look at the organization of time, tasks, and personnel in their projects. Only after the chart was drafted, refined, and accepted by the CIM/GEMS staff, and the project summary written to accompany it, did we have a common focus and consensus about what we were. The phase of project start-up was behind us, and we could move ahead into technical development. We rarely had to return to discussions about what our principal objectives were, and the two documents we produced did a great deal of work for us. We sent them out frequently to companies interested in our work. Before the documents were created, we sent our staff members instead, which took their time away from development.

**Technical Start-Up and Planning**

When we began putting a needs-analysis document together, we were exceedingly more document-driven, more accustomed to using writing as a means of problem-solving. Another trouble spot occurred for us when we were trying to identify our technical starting place, that is, exactly how the manual factory worked so that we could know how to alter it. We had to agree on what the factory actually did before we could identify what we needed to do to computerize it. Again, it is a situation of quandary, confusion, even tension, that developers must be skilled at recognizing for it signals a time when writing intervention may be needed. Trying to define the manual factory, one of the engineers began drawing on the blackboard; he then went to his desk that afternoon and came back with the diagram in Fig. 2. In this situation, the writer, like other members of the project team, responds to the "straw man" document by criticizing, refining, and restating it, not only until consensus is reached about what the document should say but until the document communicates that consensus clearly.

The only thing clear about the working document in Fig. 2 is that it is not clear, although it was clear to the developer who came up with it. The document fostered no agreement or focus for the group because everyone could see something different in it, even after the developer explained it. However, it was a good enough starting place, and our objective became to affix or embed the engineer’s explanation of the picture into it. The next version of the diagram (though there were several drafts to come between it and the original) was done on the computer and certainly looked better and was more communicative (see Fig. 3). And yet, confusion still reigns in the picture, for it is not clear what the movement suggested by the arrows is about. This tentacled box suggests something dynamic and unfixed in

---

2In this situation, the writer did not introduce the first draft of the document. Once projects become document-driven, documents will originate from other project members as well.
the manual system, yet still structured and directional. Clearly, there is connection implied between the icons at the top representing management activities and shop floor activities in the box, but what is the connection exactly? The developers still did not have a common or a consistent vision of the manual factory and how its parts worked together.

We continued to work on this document together, as we had on the organizational chart, until we were finally satisfied in
the final version in Fig. 4. We settled for an organization and iconography that showed layers of activities rather than actual movement because we could not specify on a single page all of the connections that the arrows in the second version of the graphic implied. As it turns out, everything labeled "Indirect Labor" connects in very specific ways to the "Direct Labor" of the actual production system, so we settled for a document that was more general and conceptual, and then articulated the specific connections in the function and information models we created next. The modeling, discussed in the previous article, followed a structured diagramming and documentation procedure that drove the representation of the system into levels of detail that could be used for computerization. Still, the diagram above was the consensus-building starting place we needed, an overall view of the system and its categories of activities that we would analyze further and specify precisely in models.

Writing Technical Requirements

After using the IDEF0 and IDEF1 modeling tools to identify system needs and requirements [9], we began writing a requirements document that recorded precisely what requirements the finished system would have to meet [10]. The chart we created to assign technical tasks to project members also identified the categories for which we needed to write system requirements. This chart, which appears in the previous article, provided us an outline and working Table of Contents for the requirements document. Writing a requirements document for one system is not particularly difficult. It requires mainly that lists of requirements be created for all categories of technical development. However, as mentioned before, the requirements for a custom CIM system cannot work for any other system. Thus, a document capable of more general use would have to tell how requirements for a system are identified, that is, what things must be considered, what rationale applied, and what analysis undertaken. Representing and explaining design rationale and analysis—that is, why this list of requirements and not another—is a purely narrative operation. Not only is this narrative important in the adaptation of technology, but it is important to management and end users who need to understand why the finished system works as it does, why certain limitations exist perhaps, or why there are specific constraints on its implementation and operation.

On the CIM/GEMS project, we tried to produce a document that could be used both by our development team to construct the CIM Rules Factory, as well as by persons who would use the design rationale and analysis in the document to adapt the technology and come up with custom requirements for the CIM systems they would build for their companies. Fig. 5 shows one page from the requirements document. This page is the top level of requirements for the system and justifies the general configuration we selected. Like this example page, each subsequent section discussing a technical area or part of the system presents a technical narrative that explains the specific list of requirements that follows it. By themselves,
2.1 General System Definition and Configuration

The CIM/GEMS project has established an overall system definition and configuration for the CIM Rulers Factory that will guide the work of the project in subsequent stages of system design and implementation. This section specifies what kind of CIM system the Rulers Factory will be, how the system will function and be configured, what performance will be expected of it, and how that performance will be measured and assessed.

2.1.1 System Definition and Philosophy

The CIM/GEMS project views CIM as a business strategy whereby computers are efficiently integrated and used to perform the information work in the manufacturing system. In building the CIM Rulers Factory, the project has emphasized careful needs and requirements specification before system construction is undertaken. This planning is aided by effective system-modeling tools and analysis—specifically IDEF0 and IDEF1 techniques—which identify the function and information characteristics of the system that make up its manufacturing activities.

Because the CIM/GEMS Rulers Factory has no time-critical tasks (no need for real-time control), it can be constructed as a transaction-processing system built around a relational database management system (RDMS). This is an implementation of the three-schema architecture, which permits the "decoupling" of factory workstations and their functions. Each workstation can be designed and implemented independently of the other workstations in the system, and thus be configured as a separate and self-contained unit in its own right.

As stipulated in the CIM/GEMS project proposal, the CIM Rulers Factory is a low-cost laboratory factory that will be used for CIM education and demonstration. The factory is entirely composed of personal computers (PCs) and PC software, with a hybrid production system of both manual and automatic equipment. The hardware and software selected for the factory are de facto industry standards that are readily available, uniformly configured, and inexpensive enough for even small Danish manufacturing companies. For a complete description of the CIM/GEMS project, see Appendix A, CIM/GEMS Project Description.

The following are general system-definition requirements for the CIM Rulers Factory:

1. As a requirement, the Rulers Factory will be designed as a low-cost laboratory factory for CIM education and demonstration.
2. As a requirement, the Rulers Factory will be entirely composed of personal-computer technology and software.
3. As a requirement, the Rulers Factory will be a hybrid manual and automatic manufacturing system.
4. As a requirement, the system design will be based on IDEF system-analysis models that identify the function and information characteristics of the manufacturing system.

Fig. 5. Sample page from the system requirements document.

the lists of requirements appear arbitrary. They can only be followed, not questioned. The narrative, however, substantiates and justifies decisions and supplies answers to questions that may be asked during the design phases. For example, why is the CIM Rulers Factory system a personal computer (PC) system, or why does it use the particular data it uses?

Writing technical narrative is a labor-intensive activity and, once again, may require the efforts of a full-time writer. The means of information gathering were the same as before, except that this time, specific interviews were set up with the technical staff and taped. The interviews required that the staff simply list out loud the requirements for a particular area of technical development, and then explain why those requirements were necessary or justified. The writer asked questions as needed. The text for the document was then culled, transcribed, and interpreted from these tapes. The technical staff then reviewed the document for inconsistencies or incompleteness and made changes as necessary.

Word-Processing Support for Writing

Relying on documents in this way made it crucial that we have a powerful word-processing system to support their
production. We selected Microsoft Word for the project because its “style sheet” capability accelerated both document development and assembly. (Many programs now have similar capabilities for creating formatting templates.) For example, when the example page in Fig. 5 appears on screen, codes appear on the left-hand side of the screen identifying the particular “style” or formatting that each part of the text should have. For example, a second-level heading is H2 on the style sheet (see number 8 on the style sheet in Fig. 6). Typing Alt H2 before typing a second-level heading initializes the format specifications for that text component. Everyone on the project used the same style sheet, so all documents were formatted in the same way no matter who the author was. In this way, less time was spent attending to format checking and correction.

In addition, hidden codes (single characters between periods followed by a colon) within the text in Fig. 5, which appear on screen but do not print out, instruct the program to perform special operations that streamlined the assembly of our documents. The code in front of each heading made it possible to assemble a Table of Contents for a document, a fairly standard feature in word-processing programs now. However, this same automatic table-generation feature could also be used to produce other tables. We put a hidden code before all requirements in the document, arbitrarily labeled r, so that we could compile an abbreviated document composed of technical requirements only. With a single command to compile the r table, the system pulled from the document all of the requirements, formatted them according to specifications in the style sheet, and then generated a comprehensive list of the requirements, a page of which appears in Fig. 7. This short version of the master requirements document was used by the CIM/GEMS staff, who no longer needed the design rationale and required only a checklist to follow. However, if they needed further explanation of a requirement, then the short version provided the page number on which each requirement is found in the master document (remember that this table-generation function was designed for compiling Tables of Contents, so the page number is automatically generated for each item coded to go into the table). Thus, the short list is automatically cross-referenced to the master document.

PROTOTYPING: MERGING SYSTEM AND USER DOCUMENTATION

As design requirements and specifications get pinned down and documented, developers can begin building system prototypes. Prototypes are the “look and feel” of a system, and in computerized systems, these prototypes are usually the look of the computer interface, that is, the screens that operators see at their production workstations in the CIM factory. Prototyping is the point in development when system documentation begins to address end-user requirements, and it is here that the two
2.1 GENERAL SYSTEM DEFINITION AND CONFIGURATION

2.1.1 SYSTEM DEFINITION AND PHILOSOPHY

1. As a requirement, the Rulers Factory will be designed as a low-cost laboratory factory for CIM education and demonstration.

2. As a requirement, the Rulers Factory will be entirely composed of personal-computer technology and software.

3. As a requirement, the Rulers Factory will be a hybrid manual and automatic manufacturing system.

4. As a requirement, the system design will be based on IDEF system-analysis models that identify the function and information characteristics of the manufacturing system.

5. As a requirement, the system will be constructed with a three-schema architecture, which permits the decoupling of workstation design and function.

2.1.2 SYSTEM CONFIGURATION

1. As a requirement, the Rulers Factory will be configured with eight production workstations—foreman, cutting, sanding, drilling, milling, painting, engraving, and assembly.

2. As a requirement, a PC will be located at each workstation.

3. As a requirement, the workstations will all maintain their manual process operations, except for the engraving station, which will be automated.

4. As a requirement, the workstation computers will be linked together and to a relational database management system (RDMS) and file-sharing system through a local-area network (LAN).

5. As a requirement, all production system information will be transferred via the RDMS to the appropriate workstations.

6. As a requirement, the applications used at the workstations in the Rulers Factory will have a common graphical interface.

7. As a requirement, all workstations will have barcode readers and means for data entry.

Fig. 7. Sample page from the short version of the system requirements document.

documentation efforts should be connected and merged. In fact, it is wise to begin prototyping early so that people can see where technical development is going and what it is producing. Early prototyping also gives developers more time to use the prototypes, develop them further, and consider the ways that an end user would use the interface. Like documentation, user interfaces need time for development and testing. The work invested in improving a user interface to make it more intuitive means that less documentation will be needed to support and explain it. Once again, documentation shows itself to be a part of the technical system as more of it moves online, either in the form of more self-explanatory software and screen interfaces or in help systems that support computer applications. Shirk [4], Bolter [11], and Sankar and Hawkins [12] agree that software is going to become the main medium of communication in the future. The writer’s principal tool is the computer, an important object to think with on projects, but also a door into a shared domain of networked “cyberspace” where development is done and also delivered to its users.

A simple example of a piece of early system documentation transforming itself into user documentation/software in the CIM/GEMS project is the replication of the icons in Fig. 4—the diagram of the rulers factory manual—as icons in the software interface that production operators use at their workstations in the finished CIM factory. The icons from that figure are arrayed on the operators’ screens (“Macintosh-like”), and an operator’s selection of any one of them—a work order or part drawing, for example—would bring that
piece of documentation into the screen. When a development or administrative computer system is networked to a CIM production system, as ours was (see Fig. 8), then information is even easier to share, adapt, and move among them. The stations within the gray shaded area of Fig. 8 belong to the development system and are used by the CIM/GEMS project members to administer and develop the production system. Another view of the network (LAN) and the finished CIM system from the perspective of the production system appears in Fig. 9.

CONCLUSION

The CIM/GEMS project and six parallel projects successfully completed their CIM systems. Using documentation to map and drive CIM development proved educational to our industrial coalition who used these documents as a blueprint to follow in their own projects. Thus, documentation worked simultaneously to fulfill the knowledge transfer and educational goals of the project and to assist both the internal and external projects to complete the technical tasks of CIM development. Documents also supported the effort of project coordination by functioning as an interface of explanation and appraisal between management (which often grows fuzzy about a project when the work gets very technical) and the technical developers. If management can continue to be coached through the technical development process by documentation that is regularly created during the project life cycle, then it is not so likely to diverge from the technical effort and can support it more knowledgeably with resources and within established time and budgetary constraints. A comment by an engineer from IBM Denmark in an evaluation of the CIM/GEMS project underscored the sense that our project members had about documentation and its importance.

"We don’t want to begin with Adam and Eva [sic]. The documentation discipline is a must! We consider it important and praise the project team in structuring and pushing the documentation efforts to such a high quality. We see working with language as a future tool in the arena of competition" [13].

The writing of system documentation should no longer remain the "submerged" activity that Paradis, Dobria, and Miller [14] identified as being in their study of writing at Exxon, and which Muriel Zimmerman confirms in her research of California software-development companies. Document-driven development takes writing out of the usual "nonrationalized domain that it inhabits in R&D projects where it proceeds underwater" [15] and uses it to initiate, direct, and focus technical development rather than following in its wake. The ideal situation is one in which technology works so well, performs exactly as you would want and expect it to, and is designed so seamlessly that no user documentation is needed for it at all. We still are a long way from that goal.
but, paradoxically, it may be better and more effective project
and system documentation that eventually helps us to reach it.

REFERENCES

E. Tvedt, “Document-driven management of knowledge and technol-
ogy transfer: Denmark’s CIM/GEMS project in computer-integrated
evolution from technician to technical writer,” J. Technical Writing and
00U. ICAM Project Library, Wright-Patterson AFB, OH, 1985.
Inspections, and Technical Reviews: Evaluating Programs, Projects and
[9] IDEF0 function modeling and IDEF1 information modeling are tech-
iques developed by the Integrated Computer-Integrated Manufacturing
(ICAM) project of the U.S. Air Force (1975–85). IDEF (ICAM De-
finite Method) was adapted by ICAM from SADT (Structured Analysis
and Design Technique) by Softech, Inc., and then placed in the public
domain.
McDaniel, System Requirements for the CIM Rulers Factory, Technical
Report CG08/T22, Driftaenstukk Institut, The Technical University of
ionals in large software projects,” IEEE Trans. Prof. Commun., vol. 34,
on the writing environment of an R&D organization,” in Writing in
Nonacademic Settings, L. Odell and D. Goswami, Eds. New York:
[15] M. Zimmerman, “Are writers obsolete in the computer indus-
try?” in Text, Context, and Hypertext: Writing with and for the Com-

Ellen McDaniel is a User Communication Analyst for Engineering Computer
Operations in the NCSU College of Engineering. She works on the Eos project,
which is implementing the distributed computing environment developed at
MIT in the Athena Project. She is also an adjunct faculty in the NCSU English
department and researches computer-based writing and learning environments.

Robert E. Young is an Associate Professor of Industrial Engineering at North
Carolina State University, Raleigh. His research and teaching interests are in
CIM (Computer-Integrated Manufacturing) design and implementation.

Johan Vesterager is an Associate Professor of Industrial Engineering at the
Technical University of Denmark, Lyngby, Denmark, and is an independent
engineering consultant.