CONTENTS

1

THE EXIGENCIES AND FORMS OF TECHNICAL COMMUNICATION

A fundamental challenge of organized human labor is to coordinate with others on both the concept and object of work. To assist with that coordination, we have constructed a bewilderingly large and complex array of supporting technologies and texts that orient us to our work. Out of this context, the profession of technical communication has emerged, to help accommodate technologies and texts to our situated uses. Documentation including technical manuals, procedures, and instructions has emerged mushroom-like around new technologies, as they have since the origins of the field. What has changed about technological development since the first technical manual, however, is the speed of technological development and iteration, the capacity for user customization, and the extent to which technologies have become integral to daily work. The changes are reflected in the form and content of documentation, and comparing technical manuals from the early pre-history of the field of technical communication to today reveals differences in approaches that tell of these changes in our access to and use of technology. Also reflected are changes in the rhetorical situations that we address with technology. Not only have our tasks changed, but also the ways technologies portray those tasks to us.

It would seem that technical manuals have an increasingly important role to play, mediating access to our technologies and, through them, to our work and each other. Still, few people read manuals today and the genre itself seems increasingly out of time. The reasons are complicated. Users have not become more universally adept at learning and using new technologies, as the idea of the "digital native" would have us believe. We still need help, but increasingly we are ignoring manuals because our purposes have grown beyond what manuals are capable of addressing. This book offers a consideration of what users need from technical communicators, which turns out to be much more than thorough knowledge of a technology, presented with scrupulous attention to the formal conventions of task-oriented manuals. Generating raw

DOI: 10.7330/9781607327622.c001

help content has always been something that technical communicators do, but the challenge today is to facilitate a manner by which users can interact with that content. Technical communication has always been about supplying thorough, useful, and usable content about a technology, and it still is, but documentation today may be less about generating content than it used to be. The change hinges on how we think about technologies and how we expect technical documentation to accommodate those technologies to users. As our technologies have become more ubiquitous, integrative, customizable, and connectable, they have become more difficult to document, largely because iterations of a technology vary by user, as do the configurations of technological systems, the "functional organs," that users rely on to mediate their work (Kaptelinin 1996, 50). Furthermore, the problems that shape the development of technologies defy neat categorization and description as guides for modeling and documenting user interaction.

By looking at the challenge of knowledge creation posed by rapid changes in technology and by the ways that tasks require users to rely on adaptive combinations of technologies to get work done, we can begin to see the problems with knowledge creation which have prompted this book.

Writing in 1999 about the need for designers to consider how technologies fit users' lives, Donald Norman sketched out a lifecycle of technology development. Early on, there is a gulf between what the technology is capable of doing and what early adopters hope it is capable of doing. Where the technology meets actual users, a gap forms between the technology's capabilities and the users' expectations, assuming that users encounter the technology in a context where the uses of the technology are not stringently managed. For many users of computer software, tasks grow larger than the software design is meant to support, leading to new developments in the software. Technologies (like software) continue to develop to a point where the users' needs, having remained the same, are met by the capabilities of the technology; it is a balance at which the technology does all that the users hope (Norman 1999, 32). Technical communication has traditionally helped to bridge this gap between what the technology is capable of doing and what users want it to do. When those things are the same, the gap is easily spanned. Beyond this moment, the technology continues to develop, improving efficiency, effectiveness, and ease of use but does little to build on its basic operation.

At this point, there is a turn as users begin to adapt the technology socially. They integrate it with other practices; they extend it; they

customize it; they network it with other technologies. This is not to say that users no longer need documentation and support, but rather that the technology they need support using has grown and incorporated more technologies into it. The technologies hybridize and become something more than the designers or documenters can anticipate. Technologies stop being standalone products and become parts of technological systems. For example, technologies like graphics editing software become part of a collection of tools for working on industrial design projects. Email clients become parts of project management tools. Just as important is that documentation needs change as well. As the tasks that users engage in with these technological systems go beyond simple interaction and dialogue with an interface, the focus shifts away from support to integration into a broader network of technological and human actors.

At the heart of this book is the point that these changes in technologies and our relationships to them are creating new demands for knowledge that are challenging our practices of knowledge creation achieved through traditional technical communication genres. At the same time, these demands are also opening up opportunities to redistribute the work of technical communication and reveal opportunities for new kinds of knowledge creation that technical communicators are perfectly able to deliver. In taking up this point, my purpose is to describe this redistribution of knowledge creation, understand why it is happening, and then look at the new kinds of knowledge creation demands that result. This period of transformational redistribution is not a bleak period in which the value of technical communication is diminished, but is instead a period in which that work is repositioned and expanded. Just as the field has undergone radical changes as our audiences and purposes have shifted, the field is now undergoing a similar change as our objects of knowledge creations are changing.

In corporate settings, a traditional role for technical communicators has been to create knowledge about a product, by describing how it connects with contexts and norms of use. Technical communicators create knowledge that moves in two directions: they help users come to know how a product works in support of their tasks, and they create knowledge about the contexts and models of those tasks that (ideally) feed back into the product development cycle.

Regarding the first kind of knowledge creation, supporting user tasks, the outcome of this work has commonly been user documentation of some sort and technical communicators have been central to that process. Yet that relationship is changing as more users are turning to

more immediate, personalized, and flexible forms of assistance that they find by communing with fellow users who have themselves struggled with and solved issues that users face (see Gentle 2012). In some ways, this turn toward user communities is part of an effect that Mackiewicz describes the "waning authority and influence of professional expertise" (Mackiewicz 2010a, 21). While that turn from professional expertise may be true in some contexts (e.g., online product reviews, see Mackiewicz 2010a, 2010b, 2011, 2014) in others, the role of professional expertise still retains importance. Another explanation for the interest in online user groups is that of changing tastes in user support. Rather than looking up answers in a manual or on a wiki or in some other location, some people would prefer to ask someone and to have a conversation about it—certainly there is no arguing that user forums are more interactive, quicker, and can offer more targeted assistance.

This desire for peer-to-peer support has likely been present for as long as we have had technologies we need help using. The idea of community support, building and relying on local communities is an older idea still. With the development of reliable Internet access, the idea of an online user community that offers support and companionship became a reality, although somewhat romanticized (Rheingold 2000) and not without ugly drawbacks (see Dibble 1993). Over time, discussions about the value of online communities and peer-to-peer interaction have become polarized between those who worry about users forming shallow and alienating relationships (see Dreyfus 2009; Kraut et al. 1998) and those who believe the opposite, that online communities strengthen relationships (e.g., Carroll and Rosson 2008; Katz and Rice 2002) and create opportunities for building social capital (Putnam 2000).

Acceptance of peer-to-peer support in technical communication has been acknowledged for decades. Mehlenbacher, Hardin, Barrett, and Clagett reported survey results showing that "a significant percentage of respondents indicated that they used the Internet for collaboration and for fostering relationships and a sense of community with other technical communication professionals" (Mehlenbacher et al. 1994, 213) further noting the broad range of information and immediacy of feedback chiefly among the benefits (213). The authors go on to point out that before the Internet of the 1990s there were Multi-User Domains (MUDs) and object-oriented MUDs (MOOs) that were once thought to be of value in supporting technical communication. The reasons for seeing value in peer-to-peer communities for technical support then are true today. First is the broad access to information that might not be available in one's local, offline community. Taking advantage of weak

and latent ties in one's online networks enables greater access to unavailable information and expertise (see Granovetter 1973; Haythornthwaite 2002) by jumping over the structural holes in one's offline community (see Kadushin 2012, 59–60). Second is the immediacy of interaction: online networks operate around the clock, with contributors from different time zones. And third is the social capital built up through the contributions to an online community: what one puts in one can be assured of getting back out.

Of course these qualities of online and peer-to-peer support have attended networking technologies from the start. Today, participation is greater because more people have reliable access to the Internet, and that fact alone enriches the positive qualities that made peer-to-peer help so attractive. Even so, it is not true that all users take all help queries to their peers online. People are still reading and using print documentation created by technical communicators but perhaps more for becoming oriented to a product rather than for addressing situated, complex, and uncertain task conditions that technical communicators would not be able to anticipate in writing documentation. It is in addressing user needs that are complex, situated, uncertain, and changing that online peer-to-peer support offers the greatest potential assistance. In those situations, users do want broad access to other users because understanding their help needs and understanding the solutions requires a dialogic process of discovery and refinement. Such a situation points to alterations in the rhetorical work of technical communication, beginning with the understanding that "[m]any kairotic determinants are beyond the rhetor's control, a reality that complicates models of rhetorical agency (Sheridan, Ridolfo, and Michel 2012, 7), notably the technical communicator's role as a producer of definitive knowledge about a technology. More of that work is shifting to communities of users who can collectively act as a more flexible, distributive source of knowledge. Indeed, many organizations readily acknowledge that some users prefer more interactive and dialogic means of assistance and take care to establish their own user communities (e.g., see user communities at Apple, Adobe, Microsoft, SAS Institute, etc.)

If more task support duties are delegated to communities of users, the result may be a break in the knowledge production circuit that technical communicators helped close. If technical communicators are not the ones exclusively providing knowledgeable task support then they are more distanced from the contexts and models of those tasks and become less capable of feeding knowledge back into the product development cycle. Even so, the point of this observation is not that technical

communicators have lost their place in the knowledge production cycle but that they need to shift and redistribute their efforts in order to work with communities of users. Technical communicators are needed for helping communities of users remain productive and welcoming (see Frith 2014) because when user communities are working well the patterns of questions and conversations can reveal much that is of significance for further developing a product. Conversation in communities often reveals the plasticity and complexity of user tasks and contexts and their continual change. The conversations also produce volumes of useful and sometimes reusable product knowledge.

My aim is to re-situate technical communicators as contributors to the knowledge cycles that they have traditionally been a part of. I will do so by first creating room for user communities and seeing what kind of knowledge they create through exchanges with users. What kinds of issues are compelling users to seek out help from other users? How can user communities be better equipped to engage users in consistently productive ways? What is the value of the knowledge that user communities generate and how can that knowledge be preserved, shared, and reused?

Before jumping into a discussion of user communities and knowledge producers, it is useful to consider how the role of technical communication in knowledge production has changed over the decades. Just as the lifecycle of technology development predicts a moment at which a technology becomes a socially-adapted construct (see Norman 1999) the development of technical communication shows a similar development toward more deliberate attempts to address the social and to define the technical communicator's knowledge production work in those terms.

THE EXIGENCIES OF TECHNICAL COMMUNICATION OVER TIME

Histories of technical communication trace a familiar timeline for the discipline, between the 1850s and the 1950s. The field had its origins in engineering education, with initial concerns focused on report writing and business correspondence. Writing education at the time focused on maximizing efficiency and effectiveness, across a finite variety of forms that engineers were called upon to write (Connors 1982). These technical documents were specialized and developed to support engineering work. The exigency addressed was to enable engineers to communicate with one another, assuming a shared background and context of work. The documents reinforced a vocational focus on engineering, to the exclusion of larger social concerns or contexts of use. The purpose of documentation started to shift as those in engineering and engineering

education worried more about training engineers as mere technicians and not as people who must interact with other areas of human culture and have an impact upon it (see Kynell 1999, 146).

Historians of the field (e.g., Connors, Kynell, Gould) typically agree that the years preceding and encompassing the world wars led to the development of familiar forms of technical writing, namely the user's technical manual. War preparations created a need for the technical manual because

> [f]irst, as the sophistication of weaponry increased, manufacturers needed writers to explain that technology to workers who lacked a technical background. Second, engineers, who had previously been largely responsible for writing user documentation to accompany their creations, had only a few English courses to draw upon for the challenge of explaining technology to the sometimes technologically ignorant. (Kynell 1999, 148)

The exigence was to communicate how a technology was to be operated safely and efficiently, in the absence of the engineer who designed it. The aim of the documentation was to allow users without any prior knowledge to acquire an understanding of a technology and use it within the parameters of its design.

The stage is similar to what Norman described as the early adoption phase in a technology's lifecycle. The technology was more than capable of meeting the users' expectations, as defined by their perceived needs. In a military context, where the roles for enlistees are tightly managed, the technical manual needed only to accommodate a technology to a user within the scope of his/her position. And given the hazards associated with the use of wartime technologies, instructions for exact operation, rather than guidelines for user adaptation, were recommended. The instructions were often goal-oriented: steps were laid out numerically and each action had its place in a hierarchy of actions. The aim was to "convey one meaning and only one meaning," such that a reader "must not be allowed to interpret a passage in any way but that intended by the writer" (Britton 1965, 114). The technical manual became the proxy for the engineer who designed it, and the most that document could hope to achieve was to provide instructions that "will not blow up your house or cut off your thumb" (Freedman 1958, 57). The gravest sins of technical writing at the time included fuzziness of meaning, poor word choice, empty wording, and mechanical errors, among others (Freedman 1958). Techniques of expression that were common included the use of an implied "you," use of the imperative mood, and use of the active voice. There was little use of jargon that was not thoroughly defined and illustrated. Technical communicators

created knowledge about the technology, which was not expected to vary much across situations.

Technical documentation of the time made assumptions about how users wanted and needed to interact with their technologies. These were users who were learning to use new technologies to perform roles with specific and concrete outcomes, in a finite variety of situations. Adaptation of documentation to military needs stands out as the strongest exigence from which this familiar form of documentation emerged, but the same need persists today. For example, people learning to operate machinery at work will have the same sorts of needs—they don't need, nor is it desirable for them to learn to adapt the technologies to different activities and this kind of observation generally holds throughout this overview. While I connect evolutionary changes in the technical manual to different historical events/periods, it is the case that we carry forward the same needs today. The change that I aim to highlight is that as new technology is added and developed, our needs are growing more complex. It is an argument that encompasses forms of technical documentation that have preceded what I believe we have arrived at today.

Early computer documentation from the late 1950s through the early 1970s tended to reflect more of a system's perspective, what the user should do to operate the technology within the parameters of its design, and in this way technical documentation was an extension of the same kind of knowledge production, where technical communicators strove to convey the intent of the technology designers to the users. Many of the concerns that guided documentation in the pre-war and war years also guided that for computer hardware. A difference with this exigence is that the computer users tended to have more initial familiarity with the technology than GIs might have had with war technologies. As a result, their needs went beyond basic operational knowledge to learning more sophisticated functions that the technology was capable of supporting.

Another factor was the fragility of the technology and its scarcity. Computer terminals and hardware were expensive and shared among multiple users (see Johnson-Eilola 2001). While there was still a gap between the work the technology was capable of supporting and what users were required to do, the focus of the documentation reinforced efficient and effective operation of what was available, if only to maximize the availability of that technology to all who needed to use it. Documentation helped train people to become skilled and efficient users, motivations that manifested as a proclivity for data tables, checkboxes, and standard procedures. Although some of these genred elements persist to the present day, "the dynamic exigencies of computer

use inevitably led computer operators to develop alternative, supplemental documentation" (Zachry 1999, 25) to serve a similar purpose.

At this point, a parallel development in documentation for other technologies shows what happens when documentation chases technologies that escape into and variegate in social circles. Also during the post war years, other kinds of technologies (e.g., sewing machines, kitchen appliances, etc.) started to proliferate and show up in households, creating a need for documentation that allowed for the accommodation of technologies to domestic spheres in which they were found (Lippincott 2003, 327–28). There was a subsequent need to reach that audience through a variety of symbolic and multimodal forms (331; Durack 1997, 249–50). The subject matter of technical communication opened up to a variety of domestic technologies.

At this moment, we are talking about a context that required adaptation of the technology rather than mere operation of it. Especially in these contexts, technical communicators became responsible for creating knowledge about technologies that extended into social settings and practices in which the technologies were used. Further, with some of these domestic technologies (and later software technologies), we start to see what is referred to in the social construction of technology literature as an interpretive flexibility, brought about first by changes in the context of use and later by actual flexibility of the technology itself (Bijker 2010). Here, too, the proliferation of domestic technologies only marks a significant historical moment in the development of technical documentation that persists today. For one of the first times, documentation had to address an audience of users whose skills and expectations exceeded what the technology was capable of supporting. The documentation had to accommodate the technologies to those users but not in a way that oversimplified what those users knew.

Using Durack's examples of the sewing machine and the dishwasher, we are confronted with a need for documentation to support efficient and effective use of a technology that may not meet the audience's full range of expectations. To the extent that the workings of the technologies were understandable, they could be adapted to new uses. To accommodate the expected and unexpected range of user uptake, the instructions may get less precise, relying on the user to supply more of the situated and experiential knowledge required to put the technology to use (Durack 1997, 258). There are dimensions of tacit knowledge, for instance, that are omitted from technical documentation in the form of sewing patterns and recipes. Now, the writers of technical manuals needed to account for the expanding realms in which their instructions

would be used. This expansion required setting information down about the conditions and configurations needed to carry out the instructions.

Although domestic technologies were starting to push on the form of technical manuals to include more situated guidelines for operation, efficiency and effectiveness still remained standard measures of a manual's effectiveness. In the realm of hardware and software documentation, models of documentation were also starting to reflect situated use, but the gap between what users expected and what the technologies were capable of doing was still sufficiently large that the compulsion toward situated adaptation was not as strong. In these situations, the tendency would have been to evaluate technical manuals in terms of how effectively they resulted in users performing the tasks as described (see Carliner 1997, 256–57).

The benchmarks for assessing technical manuals addressed qualities of technical manuals that might impede effective and efficient use: issues such as legibility and readability as well as measurable affective outcomes such as usability (external objective) and user satisfaction (internal subjective) (see Smart, Seawright, and DeTienne 1995). Users were thought of as inexperienced and in need of being addressed directly, so as not to impede efficient and effective learning and use of technology. This exigence resulted in recommendations for simple and direct language, short sentences, active constructions, sequentially ordered steps, and a simple focus on one item/task at a time (see Sullivan and Chapanis 1983).

By the 1980s, the computer manual was becoming commonplace, following more computers into households. One driving force behind the broad adoption of personal computers was the switch from a command line interface to a GUI interface and mouse controls that made personal computers more accessible (see Johnson-Eilola 2001). With an ever-widening base of users, encouraged by improvements in overall usability, computers started to show up more regularly in work and social settings (see Zachry 1999, 23). Software became more specialized and task-oriented. The readers' focus on instructional content shifted from simply reading to learn about the software to reading to learn to do something with the software. Around this time, the change in constraints around the consumption of computer technology created a new exigence for software instruction. And with this shift in exigence came a shift in the form of software documentation toward a model that is more recognizable today: the content was task oriented, articulated as a series of steps, and perhaps introduced by some conceptual information that oriented readers to the task at hand (Farkas 1999).

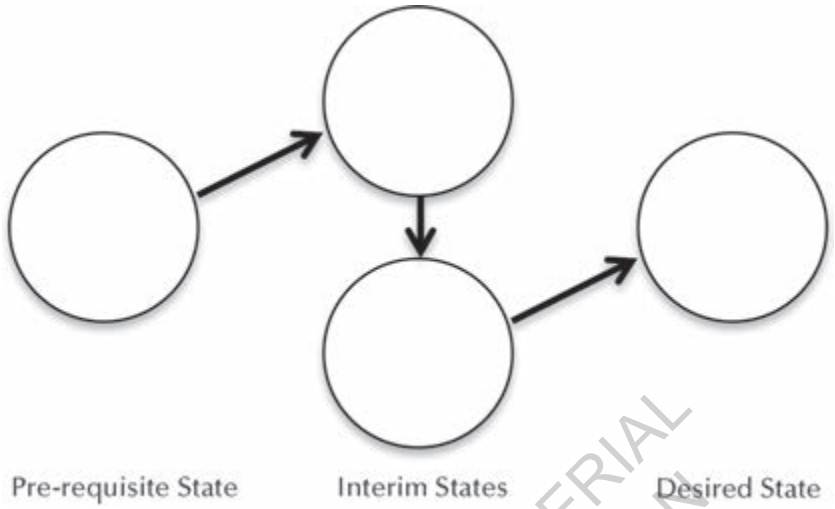

Figure 1.1. Abstract model of procedural discourse (Farkas 1999, 42–43)

"A manual mediates between the machine and users. Therefore it is essential to conceive of products as collections of uses, not a collection of features (modules calling each other). Since the only reason a product exists is to serve its users, the only justifiable way of documenting it is task oriented" (Oram 1986, 10). Even so, the suggestion followed that the purpose of the documentation was to portray what the software product was capable of doing and how that designed functionality served the user (10). The software became more specialized and task-oriented but manuals supporting use of those systems moved toward greater standardization and simplification of use (McGraw 1986).

Early on, software manuals still needed to accommodate software to users whose intentions and needs did not yet exceed the capabilities of their hardware or software. Users still needed to learn the proper ways to do their tasks, as supported in the design of their equipment. Social uses of that equipment had not yet outpaced their development. Most documents started to reflect what Farkas (1999) referred to as a logic of procedure writing (Figure 1.1) where users are assumed to start from a prerequisite state which might include assumptions about the setting, the computer system, the software version, screens, settings and the like.

Starting from the prerequisite state, the user identifies a desired state, where they hope to arrive. The route from the prerequisite state to the desired state takes the user through a series of interim states. Notable about this format is its persistence and its singularity. There is one prerequisite state and one desired state. The task is simplified and made

up of titles, conceptual elements, headings, steps, and notes (1999, 46). This genre model, which persists today has genetic relationships to the GOMS (Goals, Operators, Methods, and Selection Rules) based models of documentation (see Card, Moran, and Newell 1986). Many of the typified features of technical documentation emerge in manuals of the time, including a focus on plain and simple language, isolated focus on singular tasks, given to new development, sequential steps, and the consistent use of headings and subheadings to create a schematic representation of a task that follows the designed capabilities of the software (see Walters and Beck 1992, 165). This is a picture of a genre responding to a need for learning that is constrained.

The GOMS model attempted to decompose software tasks into sequences of actions that are oriented toward goals, fulfilled through steps (i.e., operators), in a particular sequence (i.e., methods) with occasional need for selecting alternate paths (i.e., selection rules). Documentation modeled this way resulted in unequivocal formulations for carrying out software tasks, formulations that matched the developers' sense of how tasks ought to be accomplished. Many of the genre elements that we associate with documentation today (e.g., goal statements and steps) appear at this moment and reinforce adherence to a particular model of a given task. And the role of the technical communicator here was to create knowledge of how to use software as intended, in support of user tasks that were stable and well-enough known to be modeled and directly supported. Arguably, the technical documentation was designed to maintain this optimal balance between software capabilities and user needs. When user needs grew beyond the technology, the documentation served no clear knowledge creation function.

A notable deficiency of the GOMS approach is its lack of flexibility or adaptability to complex, situated applications (see Mirel 1998). What the approach gains in standardization of instruction it begins to lose in terms of transferability. Some software packages start to attract broader use, no longer circulating only among specialized audiences and in specialized settings. A result is that the particularities of the settings begin to push on the generalities assumed by the documentation. The actions reflected in the documentation no longer match as well, or as completely, to the actions as users experience them.

The shift in focus outward, toward tasks rather than remaining strictly inward focused on the functions and capabilities of technology, reflects changes in the user base (more knowledgeable, greater diversity of purposes) and changes in the settings where a technology might be used. This shift also accompanied a growing mismatch between the kinds of

knowledge supported by documentation and the specificities of use situations. A response was the growth of minimalist documentation in the 1990s, 2000s, and today (see Carroll 1998). Often misunderstood, minimalist documentation begins with the assumption that standard, long-form documentation is not suitable for all users because the situations in which they apply their software knowledge are full of unanticipatable demands and contingencies. Documentation that best supports users in those situations is going to encourage more discovery learning and adaptation of the software to the demands of the situation. While minimalist documentation is known for its brevity and cues to readers for exploratory engagement, it is a mistake to think of those qualities as leading to trial and error learning (Carroll and Van der Meij 1998, 63). The documentation style remains task-oriented and aimed at a particular desired state, but is designed to cue the reader to apply lessons learned and to adapt them to local circumstances (see Van der Meij, Karreman, and Steehouder 2009, 271). The minimalism may have been a feature of that documentation but it also points to a knowledge vacuum where we see a mismatch between user contexts and tasks and models of those contexts and tasks implied in documentation. To some extent, users become responsible for bridging the knowledge gaps that are opening up. To address that growing gap, minimalist documentation would include more elaborated use scenarios, purpose statements, system feedback, and visualization of tasks (Van der Meij, Karreman, and Steehouder 2009, 276, Van der Meij and Gellevij 2004, 8; Mirel and Allmendinger 2004). The features were intended to help users adapt the software to the particularities of their uses, sometimes explicitly through the use of "think" or hypothetical tasks (see Barker 1992, 72).

Paralleling documentation for domestic technologies from decades before, minimalist documentation appeared to reflect a similar extension of software to various realms where situated, tacit, and experiential knowledge governed the day to day conduct of work, in ways that were not easily or adequately reflected in long-form documentation. The room in minimalist documentation to allow users to make conceptual leaps and apply knowledge locally seems comparable to the conceptual leaps found in technical manuals for dishwashers and sewing machines that, while adequately describing the operation of the technologies, left out many of the motivations and practices that remain in the heads and in the hands of users. That computer software would eventually address users who were integrating their software to an expanding range of social and professional settings is unsurprising and likely spurred by (if not a result of) developments in user interface controls.

The personal computer had developed to a point where it met the needs of early adopters and specialized users and had started to develop and become easier and more efficient to use, allowing for deeper social integration, making software and personal computers more indispensable to a variety of work practices. Despite changes to software documentation that accompanied changes in computing, the documentation remained "self-contained," "tightly bounded and controlled," "fixed, static, and absolute," "unambiguous and comprehensive" (Selber 2010, 100). It was not meeting the users' knowledge needs as they adapted and developed these technologies socially.

Self-contained documentation also clearly identifies a necessary and obligatory professional role for technical communicators, as mediators between the technology and the user. The technical communicators possess knowledge of the tasks that users ought to learn. And working from this content, the technical communicators can concern themselves more with organization and expression of the content. Their work is to manage the publications and the manner and form of expression.

As software becomes more integrated and essential to a variety of work practices, the exigence addressed by documentation continued to change. Software became so thoroughly connected to local contexts of use that the amount of documentation needed to address that range of use was too vast to produce economically or efficiently. What have changed today are the scale as well as the tools that we use. Potentially, users can become developers of their own software, even if that development is limited to the customization of a standard software package and extending or yoking its capabilities to other technologies. It is also true that some communities of users are developing their own software (as in the case of open science software). This motivation for technology development is a useful clarification of Norman's observation about the disjuncture between a technology's design and its social adaptation. What motivates social adaptation of technologies are the local exigencies that often reflect changing knowledge needs.

Characterizing this work generally is Robert Reich's concept of symbol analytic, which he describes as work that principally exchanges symbols and information. Symbol analysts are those whose work is the manipulation of data and images and text. It is symbolic and rhetorical work (Reich 1991, 177), requiring large-scale adoption of software and hardware. Out of these job settings, new skill sets emerged, which Johnson-Eilola summarizes as experimentation, collaboration across disciplines and specializations, abstraction (i.e., the ability recognize and communicate patterns), and systems thinking (in which people use

their knowledge of work technologies to construct helpful relationships that support their work (Johnson-Eilola 2005, 29–30). To engage in this work requires a person's ability to collaborate, distribute, share, and manipulate symbolic information.

As this description of symbolic analytic works suggests, the settings in which we use software increasingly resemble networks. While scholars like Barbara Mirel (1992, 1998) pointed, early on, to the importance of situated uses of documentation, the assumption was that the situations, as complex as they might be, were still understandable. This assumption comes from looking at a network situation as a snapshot in time, characterized by a particular configuration of people, technologies, and texts, in some describable relationship.

Network is not simply a noun, describing a configuration; it is also a verb, an assembling (see Latour 2005) or dynamic interaction among actors that brings about an effect. The technologies and personnel across which any given task is assembled may continually change. Some actors become important as others fade away and with those changing relationships comes a change in how we use software to maintain those systems.

This is a model of work that Spinuzzi memorably elaborates in *NetWork* (Spinuzzi 2008) and that I recall here to point out that when situations are not only multiple and complex but also living and dynamic, then the notion of tasks as things that can be captured long enough to put in print becomes problematic and so the kind of knowledge generated through documentation practices grows more disconnected from the contexts in which it is used and more out of synch with the time scale on which those uses are changing. Not only are there potentially different prerequisite states but also multiple desired states and an equally wide range of interim steps in between that may easily spill over the interface of any single technology and expand to a broader ecology of technologies and interfaces (see Johnson-Eilola 2005, 62–68). The path that one takes in a task may potentially change from one iteration of the task to the next, even within the same context of work. This change leads us back to the documentation logic that Farkas (1999) provided. A revised model would need to allow for the multiplicity of prerequisite states, interim states, and desired states that is driving the adaptability and extendibility of our software (see figure 1.2)

Here, there are multiple and changing states in the model. Each of the prerequisite states, interim states, and desired states, when examined closely, is comprised of a pulsating network of actors that regularly rotate in and out from moment to moment (Swarts 2015a). In this sense, the challenge of writing documentation for complex problem solving

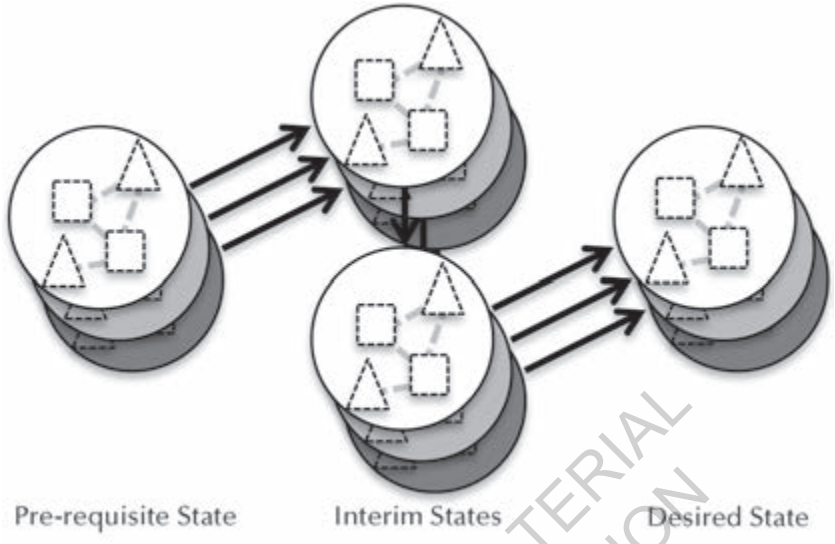

Figure 1.2. Abstract model of procedures as networks changing over time within course of a single thread

is not much different from Mirel's articulation in 1992: "we need to redefine workplace tasks and their composite computer interactions from an acting-in-situation, not an acting-with-program perspective" (Mirel 1992, 31). Erring on the side of too much stability results in documentation focused on "unit tasks" or generic tasks like "highlight text" and the subsequent assumption that a user's work consists of an accumulation of unit tasks (Mirel 1992, 11), a model that is too simple but is certainly suitable for some user audiences. Documentation that is useful in networked situations requires an extension that Mirel relates to "constructivist" documentation, which "widens task boundaries to include the social, cultural, and technological dynamics of users' work" (16) and focuses on shifting the object of instruction in documentation to the user's activity.

Consider what this change means for how technical communicators create knowledge. Addressing a user's activity through documentation requires more than an expansion of the scope of the documentation and more than the faith that users will be able to bridge from a more generic articulation of their tasks to their situated circumstances. A new approach that meets this emerging exigence will need to focus on presenting and managing knowledge (see Selber 2010, 112) as well as attention to better ways of reading situations as "complex and multifaceted contexts that are simultaneously material, discursive, social, cultural,

and historical. The struggle calls for prepared rhetors to be kairotically inventive" (Sheridan, Ridolfo, and Michel 2012, 11). In addressing these exigencies and situations, writing documentation will look different from documentation of the past and will have to include consideration of factors that arise before, around, and after the moment that compels the creation of documentation. Or it might not look like documentation at all. Instead that work may get delegated to other sources, at the same time creating new knowledge demands and needs for technical communication expertise.

Although I have characterized these shifts in technology and documentation as a linear progression from relatively rudimentary technologies and simple user tasks to complex technologies and variegated user tasks, it is more appropriate to think of these developments as points on a loop. New technologies are developed all of the time and there are still situations in which users learn technologies to support routine and simple tasks. So it is not that the forms of documentation outlined here are no longer useful; they all have their applications. Instead, the point of the historical narrative is to point out that developmental changes in the genre of the technical manual (what we call documentation today) reflected changes in the technologies that they were documenting and the exigencies that led users to adopt those technologies. My argument is that the continued development and social integration made possible because of inexpensive software, hardware, and networking capabilities, coupled with the thorough integration of technologies into our social and professionals lives (see Norman 1999) are creating a need for further change in documentation and a redistribution of knowledge creation duties to include both users who are situated and contextualized and technical communicators who can supply a different perspective on the kinds of knowledge that should be created. While these demands are relatively new, they have not gone unrecognized among technical communicators, who have worked to expand beyond standard documentation sets in order to provide more interactive and dynamic help.

The roles that technical communicators can play will vary by the type of user communities in which they participate, and so a related purpose of this investigation will be to examine two kinds of user communities: those that are organizationally sanctioned and those that are independent of organizational influence. In those user communities that are sanctioned by organizations, technical communicators may play a more visible role, advocating for best practices in response to the situations posed by users. In independent user communities, the communicator's role might be more behind the scenes, supporting the community by

facilitating creation of effective help. In both contexts, technical communicators are needed to help communities hang on to what they know, cultivate knowledge sharing practices, and create knowledge that cycles back into product development. But to get to this discussion, we will need to start with the evolving tasks to which documentation now responds. Doing so will first show why online peer-to-peer help solutions might be attractive to users and then show why user communities might be so useful in producing the kind of knowledge required to address these tasks.

The remainder of the book will develop around three key rhetorical challenges, each of which speak to some changed aspect of the rhetorical situation of technical documentation: wicked and tame problems, the decentering of expertise, and help as a social act. Each of these rhetorical challenges presents an opportunity to talk about how knowledge creation is redistributed in this age of technical communication.

In chapter 2, I take up the issue of task shift and changes in audiences and constraints. I show that the reasons people need task documentation are changing from learning discrete solutions to discrete problems, tasks that are well within the boundaries of a software's programming, to more unbounded, complicated, and emergent problems that may involve multiple software agents and a host of constraints and actors that cannot or may not be known ahead of time. In so doing, I attempt to articulate what kind of knowledge users are seeking.

Chapter 3 moves from an updated notion of task and audience to examine the issue of wicked (i.e., boundless and expansive) versus tame (i.e., bounded and constrained) problems that arise in the various networks where we use software. To get at this issue, the chapter will rely on the results of field research on forum postings for various software packages. My purpose is to show the kinds of problems that participants on those forums bring to the group and the level at which they are able to express those problems or tasks. What kind of knowledge creation activities have user communities taken up?

Chapter 4 builds on the analysis of problems and tasks in chapter 3 by analyzing the problem statements and considering the follow up from the community members. I show how expertise becomes decentered when dealing with task-shifted problems—the most notable change is that the technical communicators move away from their standing as the sole creators of knowledge. In fact, when technical communicators attempt to retain that role by sending back standard documentation solutions, their credibility suffers. The truth is that the technical communicator no longer needs to be the sole creator of knowledge. Instead,

this role can shift to the crowd, but not without problems associated with trust. Attendant to this issue is that of establishing credibility and ethos. If the technical communicators are no longer the source of credibility, granted to them by their association with a company, then wherefrom does the authority arise? How do crowds appeal to their credibility through habits of mind, habits of value, and habits of emotion? Looking at these techniques in actual examples and drawn from the same study of forums will show how the crowd legitimately shifts into a role of prominence and authority.

Through the techniques that community members use to engage with visitors to the forum, we can see a process of engaging with problems/issues/tasks as kairotic moments that create the need for documentation. I characterize this activity as a kind of crowd-based stasis work, wherein community members explore the questions and conditions that arise around an issue to develop a better understanding of the issue. This interaction is a form of help, not as an object of documentation but rather as an activity, help as an event and a particular kind of knowledge. By understanding tasks as shifted or shifting, we can see how forums or performance spaces like them become places where techniques of argumentative stasis can be used to identify issues within wicked problems that can be addressed by documentation and to find an exchange dynamic that allows the participants to work through what remains. The benefit of the community is that the conversation helps users resolve problems in ways that establish the facts (conjecture), define problems and their scopes (definition), probe the causes and mitigating circumstances (qualitative), and debate whether the forum is the right place for the discussion at all (jurisdiction).

Chapter 5 examines the outcome of iterative cycles of stasis: recurring forms or proto-genres of documentation that reveal the kinds of recurring social actions through which help is provided. The record of interactions between community members shows a process of help as an event. The threaded record of that interaction then stands in as a trace, a genre record—it describes help activity that has taken place and creates expectations that guide future interactions. The threads themselves are not always used or accessed as static documentation, but instead capture the interaction between community members through which help is provided. I consider four different kinds of recurring social actions that provide help in the moment but that are tailored to fit the particularities of their situations. I offer and discuss proto-genres including work throughs, work arounds, best practices, and diagnoses. Through this discussion, I can (finally) discuss how the delegation of

knowledge creation to user communities creates different kinds of meaningful knowledge creation tasks from the organization and storage of community knowledge to communication of that knowledge and experience back into the technology production cycle.

Finally, chapter 6 expands on the changing role of technical communication and looks at how the redistribution of knowledge creation to user forums results in new kinds of knowledge demands that technical communicators ought to be addressing. Moreover, I make the argument for looking at a broader range of technical communication practices, at the range of ways that we have always participated in knowledge production from generating raw help content, to providing guided user assistance, to evaluating user experience, to structuring and providing access to information. The contributions that technical communicators can make is to (a) facilitate conversation, to engage community members in a process of stasis whereby topics develop into issues; (b) help the community act like a community and value each other in the process; (c) encourage systems thinking—think in terms of systems or flows of tasks and issues and the parties that need them; (d) structure information and keep it schematically organized, well labeled, and findable, and finally (e) cycle the knowledge generated about users and their contexts back into the technology development cycle, reasserting a function of the technical communicator as articulator.