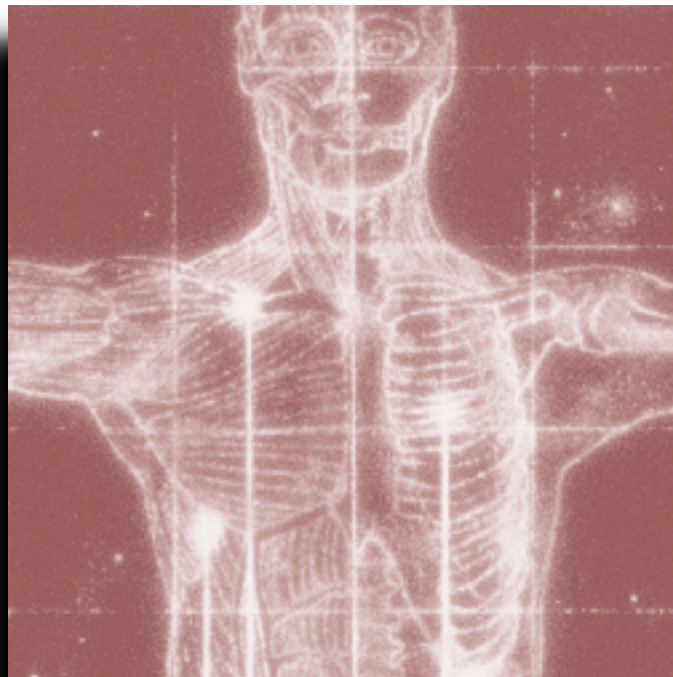


Customer-Developer Links *in Software Development*



Many of the best ideas for new products and product improvements come from the customer or end user of the product [15]. In the software arena, tapping into this source of information requires the establishment of one or more *customer-developer links*. These links are defined as the techniques and/or channels that allow customers¹ and developers² to exchange information.

Today, a number of factors have combined to make a wide variety of links available to software developers. Technological advances and cost reductions in the area of telecommunications have brought about the widespread use of telephone support lines and electronic mail. At the same time, the creation of alternative approaches for require-

ments determination have led to widespread use of prototyping and facilitated teams. Finally, commercialization of software has fostered new structures for customer participation through the use of focus groups, user groups, and trade shows.

The tremendous variety of links that are available today represents both an opportunity and a challenge for software development managers. An opportunity exists in that it has never been easier to obtain input from customers. The challenge is that development managers are now faced with a bewildering array of customer-developer links from which to choose.

Deciding on the number and type of links to use is anything but straightfor-

Mark Keil
Erran Carmel

¹We use the term "customer" to include "user" throughout the rest of the article. See [6] for a discussion of the term "user."

²We use the term "developer" to include the various individuals who are directly involved with the design and production of software code.

ward. In the absence of any information on the number of links to employ or the effectiveness of various links, how is a development manager to decide? This article begins to address these issues by focusing on the links that are currently in use and the experience that development managers have had in applying them.

By focusing on links that are currently in use, our goal is to draw practical insights concerning the type and degree of participation that should be used to engage customers in the development process [8]. We do this by comparing and contrasting the links that are used in two very different environments—the *package* development environment (in which software is developed as a product for external sale) and the *custom* environment (in which software is either developed in-house or under contract and is intended for internal use). Table 1, which is partially based on Grudin’s [6] observations, highlights some of the key differences between these two development environments (also see [3] for a related comparison). Given these differences, one would expect to find differences in the links that are used across the two environments.

We purposely chose to investigate these two very different environments for three reasons: (1) to obtain the broadest possible view of the links that are in use today, (2) to understand more about how managers obtained customer input in the two environments and the extent to which their environments shaped their perceptions and choices of links, and (3) to draw attention to links that are currently used in one environment that might be usefully transferred to the other.

Background

It has long been recognized that customer-developer mutual understanding and user participation are important factors in the successful development and implementation of systems [4, 8]. Prominent approaches for encouraging customer participation include sociotechnical systems theory (STS) and participatory design (PD) [1, 11, 12]. Thus, the issue that software development managers must grapple with is not *whether* customers should participate in the development process, but *how* they should participate [10].

The literature on customer participation has developed along two conceptual levels. The macro level acknowledges that user participation is beneficial (e.g., [7]) and provides general approaches for achieving this objective (e.g., PD and STS). The micro level focuses on specific requirements analysis techniques. A wide variety of specific techniques (such as those documented in [2]) now exist to aid in the process of determining customer requirements. What the existing literature misses is the middle ground that exists between these two levels, where managers are faced with the dilemma of choosing a combination of links that can be used to exchange information with their customers.

Therefore, our focus is not on specific requirements analysis techniques per se, but rather on the combinations of techniques and communication channels that are used in practice to establish linkages between customers and developers. The term *customer-developer links* is used here to describe the

Table 1. Relevant Differences Between the Custom and Package Development Environments

Development Dimension	Custom	Package
Goal	Software developed for internal use (i.e., usually not for sale)	Software developed for external use (i.e., for sale)
Typical point at which most customers are identified	Before development begins	After development ends and the product goes to market
Number of customer organizations	Usually one	Many
Physical distance between customer and developer	Usually small (e.g., customers are in same building as developers)	Usually large (e.g., customers are thousands of miles from developers)
Common types of projects	New system project; “maintenance” enhancements	New products; new versions (major and minor)
Terms for software consumer	User; end user	Customer
Common measures of success	Satisfaction; acceptance	Sales; market share; good product reviews

Table 2. Customer-Developer Links

Link	Description	Commonly used in: P=Package C=Custom
Facilitated team	A facilitated, structured workshop with customers (e.g., JAD) that is typically used to elicit requirements.	C
MIS intermediary	One who defines corporate customers' goals and needs to designers and developers.	C
Support line	The unit that helps customers with day-to-day problems (also known as customer support, technical support, help desk).	P/C
Survey	Textual surveys administered to a sample of customers.	P/C
User-interface prototyping	Customers are exposed to a demo, or early version, to uncover user-interface issues.	P/C
Requirements prototyping	Customers are exposed to a demo, or early version to discover system requirements.	P/C
Interview	One-on-one with end-user; open-ended or semi-structured.	P/C
Testing	New requirements and feedback stemming from testing. Does not include bug detection.	P/C
Email/bulletin board	Customers post problems, questions, and suggestions to a bulletin board or through email.	P/C
Usability lab	Specially constructed labs for taping and measuring user subjects at work.	P/C
Observational study	Customers are followed for an extended period to learn what they do (e.g., ethnographic, protocol analysis).	P/C
Marketing and sales	Representatives regularly meet customers (current and potential) to listen to suggestions and needs.	P
User group	Customer groups convene periodically to discuss software usage and improvements.	P
Trade show	Customers are exposed to mock-up or prototype and asked for feedback at a trade show.	P
Focus group	A small group of customers and a moderator are brought together to discuss the software. Discussion is loosely structured.	P

Table 3. Overview of the Sample

P=Package C=Custom	Description
C1	Telecommunications company (listed in Business Week 1000)
C2	Large computer company (listed in Business Week 1000)
C3	Major airline (listed in Business Week 1000)
C4	Major hotel chain (listed in Business Week 1000)
C5	Medium-size beverage producer
C6	Large manufacturer of electrical products
P1	Fast-growing software tool developer
P2	CASE tool developer
P3	Fast-growing programming environment developer
P4	Small, successful programming environment developer
P5	Well-known producer of Unix tools
P6	Software division of large hardware vendor—develops office software
P7	Graphics software developer
P8	Established financial software developer (listed in Software Magazine 100)
P9	Established manufacturing software developer (listed in Software Magazine 100)
P10	Established office automation developer (listed in Software Magazine 100)
P11	Software division of large hardware vendor—develops financial software

many ways in which customers and developers exchange information during the development process (similar to [13]).³

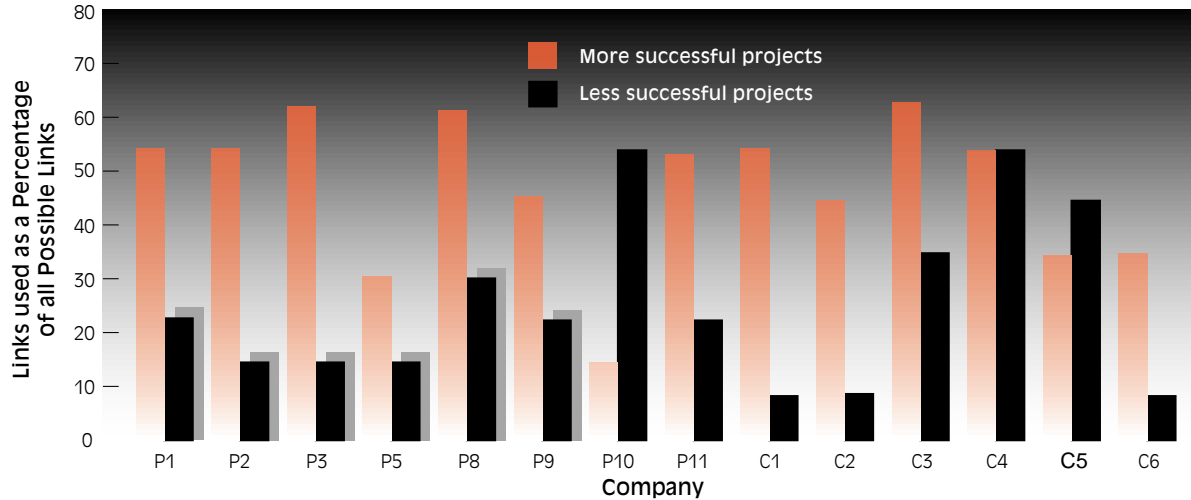
A link inventory of the commonly used customer-developer links was created based on the knowledge and experience of the authors coupled with a search of the information systems and marketing literature pertaining to customer involvement, requirements determination, and product innovation. Based on these sources and from validation that occurred during the study, the list is believed to be fairly comprehensive. Table 2 presents the customer-developer links that were included in the study, along with an indication (based on our own experience) of the environment(s) (i.e., package, custom, or both) with which the link is normally associated.

The customer-developer link inventory (of Table 2) is the foundation for a central concept of this article, namely, that these links can be *counted* within a given project. The notion of counting links has intuitive appeal because it is widely believed that greater customer participation can lead to more successful software projects. Additionally, from the literature on employee participation in the workplace, we know that participation is most effective when a combination of different approaches are used to involve workers [9]. Thus, we argue that defining and counting links is an important first step toward quantifying the domain of customer participation. We emphasize, however, that the absolute number of links is only a partial measure of customer participation and involvement—the link characteristics and how the link is employed in practice (e.g., how well the information is conveyed) may well be more important. As with any metric, counting links is but a partial measure and cannot tell the entire story.

Therefore, we augment the link inventory by introducing an important classification—that of *direct* links versus *indirect* links. From a communication perspective, direct contact between customer and developer is preferable to indirect contact because it decreases filtering or distortion that may occur. Furthermore, media richness theory [5] suggests that direct face-to-face channels offer the prospect of richer communication because of the ability to transmit multiple cues (e.g., physical presence, voice inflection, and body language). Thus, direct links are likely to be particularly important when there are high levels of ambiguity, a situation that is especially likely to occur in the communication of system requirements.

Indirect links are those in which the customer and developer do not deal directly with one another but communicate through intermediaries or customer

³Customer-developer link connotes both a channel (i.e., a medium for communication) and a technique (i.e., a method for communication). From a theoretical perspective these two dimensions may be separated, but from a practical standpoint we chose not to do this because we found that development managers themselves do not distinguish between the two dimensions.



Note: The total number of links used in each project expressed as a percentage of the total number of links possible. The denominator was normalized for the number of possible links in each category (13 for package development, 11 for custom development, as indicated in Table 2). Data were available for 14 paired cases involving a relatively successful and a relatively unsuccessful project at each company (i.e., 28 projects in total).

Figure 1. Customer-developer links used: More successful projects vs. less successful projects

surrogates. Intermediaries are entities situated between customers and developers, while customer surrogates are entities that are not true customers but are treated as such for the purposes of gathering requirements and feedback. Some links are inherently indirect because they do not typically allow for direct contact between customers and developers. The marketing and sales link (in which a salesperson serves as an intermediary) is one example of this. In other cases, the distinction is contextual and depends on how the link is actually employed. User-interface prototyping, for example, may be conducted with or without the developers being present.

Methodology

Our study followed an inductive, multiple-case study approach in which in-depth information was collected on actual software development projects. In 1994 we visited 17 companies and collected data about 31 different projects. Although the companies selected represent a convenience sample we actively sought out companies, within each environment, that represent variation along the dimensions of industry, application area, and company size (Table 3 provides brief descriptions of each firm). Thus, we believe that there are enough companies and enough variance in the sample to make reasonable inferences about the range of methods that are currently being used to obtain customer input.

At each company, a project or development manager was selected as the primary contact for the study. These managers were chosen (over customers) as the primary subjects because of their ability to comment

on the full range of customer-developer links used during the development process.

Each manager was asked to select two projects that he or she had managed (or been closely involved with) within the past few years. In selecting the projects, the managers were instructed to pick one project that was relatively successful and one project that was relatively unsuccessful. Since there are many definitions of success, we left it up to the development manager to choose both the criteria for success and failure and the project in each case. This approach served two research objectives. First, it ensured that there would be some variance with respect to the relative success of the projects in our sample. Second, it allowed us to sample paired cases in which organizational context and some of the other factors that might affect the success of a project were effectively controlled.⁴

The interviews were based on a structured interview guide and survey focusing on the 15 different customer-developer links presented in Table 2. The interview guide included a page of definitions describing all of the links. These definitions were discussed with the respondents during the course of the interview thereby increasing the reliability of the survey. All interviews were conducted by the authors and each one lasted approximately two hours. The interviews were tape-recorded and then later transcribed. Interview transcripts were analyzed using a variation of the pattern-matching technique advocated by Yin [16].⁵

⁴Three of the package firms do not have a “less successful” project pairing because either the manager could not suggest one even after repeated probing, or because of interviewee time constraints.

Results and Lessons

Having described the basic approach used to collect and analyze the data, we now turn our attention to the results of the study. We present the results in the form of three lessons for software development managers. The first and second lessons are based on two observations that held across both environments. That similar findings emerged across such different environments suggests that the lessons are likely to be robust and applicable to a wide range of environments. The third lesson is a speculative one based on differences that were observed between the custom and package environments.

Lesson #1: The More Links the Better...

Up to a Point

The prescription that more links are better is based on a strong correlation that was observed between the number of links used and the relative success of the project, as shown in Figure 1. While correlation does not necessarily imply causality, we suggest that

the 14 successful projects was 5.4 compared to a mean of 3.2 in the 14 less successful projects. This difference was found to be statistically significant in a paired t-test ($p < 0.01$). Additionally, a separate paired t-test for both package and custom projects revealed that the same pattern held across each environment ($p < 0.10$).

The interviews with development managers offered corroborating evidence of a relationship between a project's outcome and the number of links that were used. At the end of some of the interviews, managers were asked to comment on whether there was any connection between the number of links and the relative success of the two projects they had described. In several instances, the managers felt very strongly that the dearth of links in the less successful projects was a significant factor in explaining the less-than-favorable outcome. In other cases, as one would expect, there were other reasons offered as to why projects were judged to be less successful.

Finally, the less successful projects suffered not only from a low number of links but from a low number of *direct* links: 10 of the 14 less successful

projects involved either zero or just one direct link between customer and developer. As a result, we speculate that having multiple direct links may be particularly important to the successful communication of customer requirements and hence to project success.

Determining How Many Links to Establish. An intriguing issue that arises from our study is to determine the minimum threshold of links a project should have and the presumed point of diminishing returns. Based on our data, we suggest that managers should establish at least four customer-developer links, but that by the time one reaches six to seven links, there is a point of diminishing returns.

First, we examine the lower boundary: since the mean number of links that were observed in the less successful projects was roughly three, one can infer that there may

be a threshold above three associated with the number of links needed to ensure adequate communication between customers and developers. As a rough guideline, managers should probably regard four links as a safe minimum.

Next, we examine the upper boundary. While Lesson #1 states that more is better, the law of diminishing marginal returns suggests that having too many links may be sub-optimal. While each new link adds more information, the marginal value of that information drops off as the developers become more aware of the customers' requirements (see Figure 2). Since the value of additional information is intangible, in practice it is not feasible for the manager to quantify the precise point at which marginal benefits equal marginal costs. Nevertheless, the data obtained

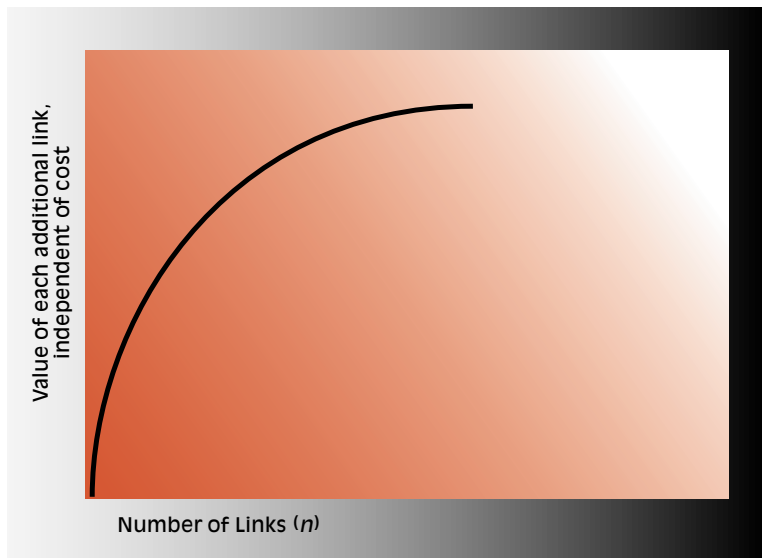


Figure 2. Establishing too many links may be sub-optimal

managers should err on the side of providing more, rather than fewer, links whenever possible.

Results of the analysis revealed that in 11 of the 14 paired cases, the more successful project involved a greater number of links than the less successful project. The mean number of customer-developer links used in

⁵The package cases and the custom cases were analyzed separately to look for within-group patterns. Within each of these groups, the more successful cases were then separated from the less successful cases and the pattern matching process was continued. Finally, the patterns that were observed in the package cases and the custom cases were compared to see if there were any cross-group patterns that emerged.

from the projects we studied provide some indication of when the law of diminishing marginal returns is likely to occur. We observed that even the successful projects used a relatively small fraction of the available links. In fact, very few of the 31 projects in our sample used more than 60% of the possible links that can be used to obtain customer input (see Figure 1). The more successful projects used an average of 5.6 links (s.d.=1.8, $n=17$). This suggests that the costs associated with implementing additional links may exceed the benefits, once we reach a half-dozen or so links.

Lesson #2: Reduce Reliance on Indirect Links: Intermediaries and Customer Surrogates

Having discussed the number of links in Lesson #1, we now turn to an issue concerning the use of direct versus indirect links.⁶ Many of the development managers that were interviewed perceived that the problems associated with less successful projects resulted, at least in part, from over-reliance on intermediaries or customer surrogates. It is on this basis that we suggest managers reduce their reliance on indirect links, substituting direct links between customers and developers where possible.

There are at least two reasons why intermediaries and surrogates are poor substitutes for direct links between customer and developer. First, intermediaries can intentionally or unintentionally filter and distort messages. Second, intermediaries may not have a complete understanding of customer needs. The following comment from one of the development managers illustrates the latter point:

The person who helped us define the requirements was an MIS intermediary who had been involved with the programming of [another application on the same hardware] in a different area of the business. From a usability/functionality standpoint, the MIS intermediary didn't have much knowledge...she wasn't a very good user [emphasis added] because she didn't understand the complexities of what they were asking for.

As Grudin [6] observes, these “go-betweens or mediators often discourage direct developer-user contact...and are often ineffective conduits,” particularly for information involving new requirements. In the context of surrogates, Grudin suggests that links such as user groups will be less effective when meetings are attended by “buyers rather than users and by

⁶As mentioned earlier, indirect links are those in which the customer and developer do not deal directly with one another but rather through intermediaries or customer surrogates.

Table 4a. Custom Projects: Mean Rating of Commonly Used Customer-developer Links for More Successful Projects* (1=very ineffective; 5=very effective)

Customer-developer Link	Mean Rating	Number of Projects
Facilitated Teams	5.0	4
User-Interface Prototyping	4.0	5
Requirements Prototyping	3.6	5
Interviews	3.5	4
Testing	3.0	3
MIS Intermediary	2.8	4
Email/Bulletin Board	2.5	3

Table 4b. Packaged Software Projects: Mean Rating of Commonly Used Customer-developer Links for More Successful Projects* (1=very ineffective; 5=very effective)

Customer-developer Link	Mean Rating	Number of Projects
Support Line	4.3	8
Interviews	3.8	6
User-Interface Prototyping	3.3	3
User Group	3.3	4
Requirements Prototyping	2.8	4
Testing	2.8	6
Marketing and Sales	2.8	9
Trade Shows	2.5	4

*Each link was rated for its effectiveness by the interviewee—measured on a 1–5 scale—in terms of “helping to improve the software product” (with 1 being very ineffective and 5 being very effective). Data were available for ten successful packaged software projects and six successful custom software projects. Links that were used in fewer than three projects were excluded from the analysis in order to minimize distortion in the averages. Nevertheless, the means should be interpreted with caution given the small sample size. The less successful projects were excluded from this analysis because project managers had a tendency to downgrade a customer-developer link if it was associated with a less successful project, creating a bias.

marketers rather than developers.”

In the remainder of this section, we present data in support of Grudin’s observations and offer numerous examples to illustrate some of the extraordinary number of ways in which indirect links can manifest themselves in actual practice. Based on this evidence, we recommend that managers become more aware of differences between direct and indirect links and act to increase the relative effort devoted to direct links.

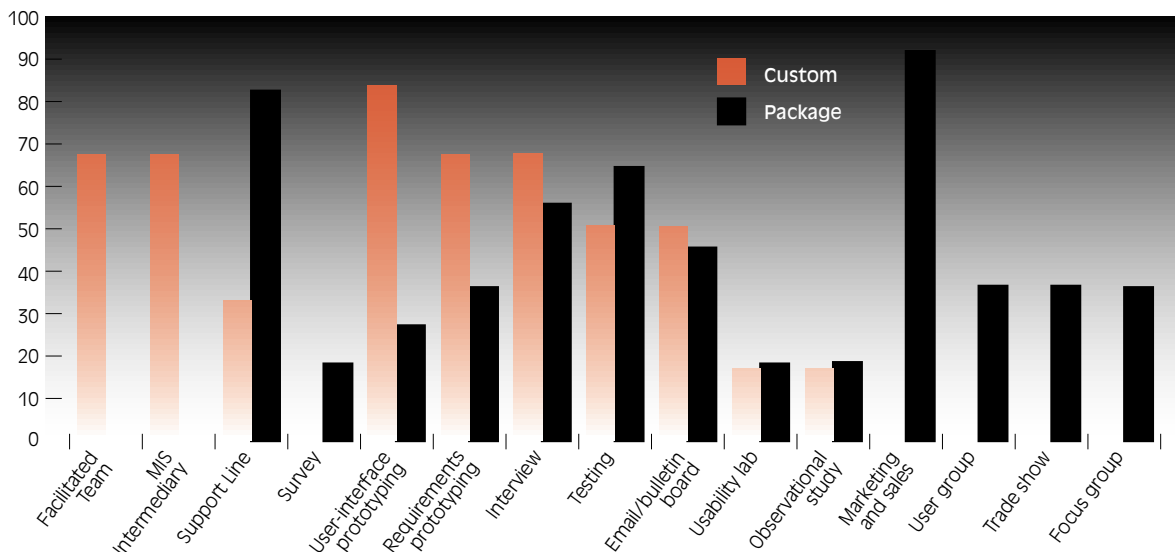
Intermediaries and Surrogates Are Widely Used but Poorly Rated. In both the package and custom environments, development managers spoke of the problems they had encountered in relying too heavily on intermediaries or customer surrogates. In many cases, the use of such indirect links was viewed as a significant factor in explaining why projects failed. The following remark made by a development manager at C2, a large computer company, illustrates this point:

The problem we had was we didn’t go out and survey the people in the field, the end users...If we had gone right to the end user and not relied on these intermediaries, I think it would have made a difference [in the outcome of the project].

Paradoxically, despite the possible problems that can result from the use of intermediaries and surrogates, the results of our study suggest that they are frequently relied on in both the package and custom environments—a finding that is consistent with previous observations made by Grudin [6]. In the custom software environment, for example, MIS intermediaries were used in 7 of the 12 projects in the sample, despite, as we will discuss later, the fact that this link was rated quite low in terms of helping to improve the software product (see Table 4a). In the package environment we also found a surprisingly heavy reliance on indirect links. In some cases, elaborate webs of intermediaries—as many as six layers—were observed between customer and developer. Over 90% of the projects relied, at least in some way, on the marketing and sales link for customer input and requirements; but this link was rated “poor” by the development managers, partially due to the traditional antipathy between the technical and sales functions found in many firms.

Intermediaries and Surrogates Can Take Many Forms. The landscape of customer-developer links is littered with intermediaries and surrogates who can take many forms. Eight examples from our data (including cases from both the package and custom environments) are described here; three involving intermediaries and five involving the use of customer surrogates. While we suspect that intermediaries and surrogates can take forms other than those that are described here, we offer these eight examples as a starting point to aid managers in identifying and reducing their reliance on intermediaries and surrogates.

- *Systems analysts as intermediaries.* At C2, a large computer company, the facilitated team (JAD) sessions were populated by systems analysts rather than actual end users. The project manager explained:



Note: Occurrences of customer-developer links presented as a percentage of the projects for each category (package and custom; n = 11 and n = 6, respectively).

Figure 3. Customer-developer link selections: Percentage of all (successful) projects (in each category) that used a particular customer-developer link

Instead of bringing all the users to a JAD session, systems analysts were sent out to the field so that they could come back and represent a cross-section of the users. That's basically the way [our company] runs. I don't think in my career at [C2] I've ever dealt directly with the end user. There's always been some type of intermediary [emphasis added]. If it were my decision, I would have gone directly to the end users.

The irony in this example is that the project team took what would otherwise serve as a direct link that encourages rich communication between customer and developer and transformed it into an indirect link.

- *Technical support personnel and VARs as intermediaries.* At P10, an established package developer, there was a reliance on formal channels for establishing links to the customer. The development manager at this firm noted that he relied heavily on the “distribution chain,” made up principally of value-added resellers (VARs) and the “support chain,” representing the support line link. The development manager noted that:

It would be pretty unlikely that we'd go and find a real end user [emphasis added] and ask them what ought to be fixed—there are just too many. I just attended a product launch the other day and got accosted by a number of customers who tried to buttonhole me to get their problems fixed. That's not the formal way.

- *Internal consultants as intermediaries.* P9, a developer of complex enterprise-wide packages, has, like P10 in the preceding example, a formalized set of mostly indirect links to their customers. In order to support implementation of their complex products, a separate services division within the company operates much as a consulting firm and bills for its time accordingly. The development manager explained:

When it comes down to design work, we really consider our consultants as users, because they have much more contact with our users than developers themselves would.

- *External consultants as surrogates.* At C4, a major hotel chain, a less successful project involved the development of a new conference management system. The system was designed to support the company's worldwide conference involving all of the company's franchisees. The problem was that the company had never managed the conference in-house; they had always contracted with an outside company that handled all of the conference management tasks. As a result, the employee who was designated to run the conference in-house—and who would ultimately use the system—had little or no experience in managing the conference. This meant that the developers had to get the system requirements from the consultant who had

managed the conference in previous years. The net result was a system that did not satisfy the customer. As the development manager recalled:

[The consultant] served as a surrogate user in all our meetings. [The person] who would become the end user of our system was at most of the meetings but she was kind of useless to us in terms of giving us requirements because the year before she had just done whatever this guy had told her to do.

- *Non-representative customers as surrogates.* At C3, a major airline, a less successful project involved the development of a new system to support ticket and gate agents. In determining the requirements for the new system, the project team focused exclusively on international agents, even though they knew that the system would eventually be used by both domestic and international agents. Unfortunately, the international agents served as poor surrogates for the much larger population of domestic agents that the system was also supposed to support.
- *Supervisors as surrogates.* This may be the most common mistake in the custom environment. We offer two examples. At C6, a large manufacturer of electrical products, a less successful project involved a customer support system that facilitated the centralization of distribution facilities. In this case, the developers were instructed by management to gather requirements only from distribution center supervisors rather than the workers who would actually be using the new information system. Ultimately, this decision meant that the requirements were never adequately assessed even though the development team emphasized the use of what would normally be seen as an effective direct link, namely facilitated team meetings. The development manager explained:

We had union issues to deal with. We were actually shutting down shipping facilities and consolidating them into one distribution center. Plants were losing certain jobs. It was all very hush hush... a secretive project. So the core group [of supervisors] that continued to meet was instructed to keep this under their hat and not to let it out [to the workers]. Unfortunately, we never involved the people who would be using the system. They were not aware of the project and there was no ability for them to come back and say: "Hey, you haven't thought about this or that." It was shoved down their throats.

At another company, the project manager explained that a less successful project “got into trouble” because individuals who had been elevated to staff positions were interviewed instead of actual end users:

The [surrogates] were people who used to be [in the field] and had now moved into a staff position ... They were supposed to represent the needs of the users to the developers. The

thing that's a little bit concerning about this is that some of these people were three to five years removed from that function.

- *Marketing personnel as surrogates.* At P11, a software division of a major computer hardware company, the marketing staff insisted that they knew the customer requirements and demanded that the developers obtain the requirements from them rather than directly from the customers. The development manager explained:

Our program management [marketing] people drive product development and they said: "We'll define the product for you." These are marketing people who have been in the ranks for a number of years, believing they know the product and understand customer requirements. They basically said: "Here are your requirements—build a product." It was a dismal failure.

There was little or no customer involvement up front. I told them [marketing] it was the wrong way to go and that it was not meeting the primary objectives of the program. The response was: "Then we'll change the objectives." The marketing pressures were so great for this application that it forced us to do the wrong things. I don't think they [marketing] understood what the product should be.

- *Developers as surrogates.* P4, a small company that produces software tools, developed a design philosophy that can be stated as follows: since the developers themselves routinely use the tools that they sell, they are their own customers. Therefore, the developers relied on themselves as surrogate users. The development manager explained:

The business of eliciting requirements from customers is very difficult. If the [requirements] are your own requirements, it's a lot easier. Your understanding outruns that of your customer. Eliciting requirements was straightforward because we were our own customers.

The preceding examples highlight some of the many forms that intermediaries and customer surrogates can take and suggest some of the problems that can result from relying too heavily on indirect links.

Lesson #3: Consider Links Not Traditionally Used in Your Environment

Lesson #3 is based on differences that were observed between the package and custom environments in the *types* of links most commonly used and in managers' *perceptions* of which links were most effective.⁷ Specifically, our data suggest that the two environments not only rely on different links but

that some of the links perceived to be most effective are used almost exclusively in one environment and not the other. We first present differences between the two environments and then offer a set of interpretations as to why these differences exist and what implications they hold for development managers. Based on the data, we suggest that development managers in each of the two environments we studied should consider using links not traditionally used in their environment but which are perceived to be particularly effective.

Differences in the Types of Links Selected. Figure 3 shows a distinct difference in the use of particular links across the two development environments (as suggested by the "P" or "C" classification of Table 2). In the custom software environment, prototyping (both for requirements and for user interface), facilitated teams, interviews, and MIS intermediaries were all used quite commonly, appearing in more than two-thirds of the custom projects. In the packaged software environment, marketing and sales and support line were the most commonly used links between customers and developers. A number of links were used in both types of project environments. For example, some form of prototyping (either for requirements or for user interface) was used in more than two-thirds of both the package and custom projects. Interviews and testing links were also commonly used in both environments.

Differences in Perception of Link Effectiveness. In addition to the differences in link selection between the package and custom environments, there were also significant differences in perceptions of link effectiveness, as shown in Tables 4a and 4b. For the custom projects the two highest-rated customer-developer links were facilitated teams and user-interface prototyping. These were the only links that received a mean rating of 4 or higher. For the packaged software projects, the two highest-rated customer-developer links were support line and interviews, with the former being the only link that received a mean rating of 4 or higher.

Most interesting, perhaps, is that in each environment there is a "favorite" link that is hardly, if ever, used in the other environment. The highest rated customer-developer link for custom development projects—facilitated teams—is not used by package developers at all. Similarly, the highest rated customer-developer link for packaged software projects—support lines—was seldom used for custom projects.

It is particularly interesting to note that in the custom environment, the four links rated most effective are all direct links. This was not true for the package environment, where two of the four links rated most effective were indirect links, namely support lines and user groups.

⁷Interestingly, there was no statistically significant difference in the absolute number of links used in package vs. custom software projects; the mean number of customer-developer links used in the successful packaged software projects was 5.7 as compared with a mean of 5.3 for the successful custom software projects.

Explaining the Differences Between the Two Environments. We suspect that the differences in link selection and perceived effectiveness can be largely attributed to structural differences in the two environments. Indeed, given the characteristics of the custom environment noted in Table 1 (e.g., relatively small number of customers that are often co-located with the developers, and who can be identified before development actually begins), it is not surprising that a link such as facilitated teams can be used effectively.

By the same logic, it should come as no surprise that the support line link was the second most commonly occurring link in the package environment and the one rated highest (in effectiveness) by development managers in this environment. This is consistent with previous studies [13, 14] and underscores the important role that support lines are believed to play in the development of packaged software. As noted earlier, development managers in the package environment are often confronted with a much larger and more geographically dispersed customer base, sometimes making it impossible to identify customers in advance of development. The support line provides a cost-effective way of reaching such a customer base.

The differences in link selection lead us to the question of what is the implicit strategic objective of link selection. Can these differences in link selection be explained, as we have done, predominantly by the dispersed versus co-located customer base? We posit that there is another explanation—that what we observed is a case of *market fit*. Package firms fit their customer-developer links to support ongoing enhancements to existing products (new releases) since their objective is to extend their product life-cycle by constantly upgrading and adding more functionality. At the same time, information systems organizations developing custom applications tune their links for initial development with comparatively little emphasis placed on developing an ongoing relationship with customers. This may be a legacy of the separation between new system development and so-called maintenance. The very term *maintenance* is misleading, yet it may signal an important distinction that exists between the package and the custom environment with respect to ongoing enhancement to existing software and the corresponding need to establish links to support this activity.

The findings for package link selection present a paradox that begs another question: How do packaged software organizations successfully use support lines—which might otherwise be viewed as a less

desirable *indirect* link—as a principal conduit for information? We offer two explanations. First, direct links may not be as necessary for conveying information associated with enhancements to existing products, an area of great concern for packaged software developers.

Second, and more interestingly, we believe that the answer lies in what we observed about how these firms have elevated the status and relationship of support personnel vis-a-vis development personnel. Many of the development managers in our sample spoke of their significant investment in support line personnel, the high levels of trust they had in them, their knowledge of the products, and in their ability to collect customer input. For example, at both P1 and P3 (both packaged tool developers) the support personnel maintained very close continuous contact with the developers and were viewed by the development managers as an extension of the development team.

Implications for Development Managers. Given the differences observed between the two environments, and our interpretation of why those differences exist, we believe the lesson for development managers is to think broadly in selecting possible customer-developer links. While some links may be specific to a particular environment (e.g., trade shows), there is no reason why other links would not be transferable across the two environments that were studied. Therefore, we believe that development managers would do well to consider using links that have evolved outside of their particular development environment. Specifically, we recommend that development managers in the package environment consider using various facilitated team techniques as a direct link that may be particularly useful in the development of new software products or major enhancements to existing products. Similarly, we recommend that development managers in the custom environment focus more attention on using support lines of various types, as an indirect link to support “maintenance” of existing software. As can be learned from the packaged software environment, this strategy involves elevating the status of these units and individuals.

Conclusions

To summarize briefly, the contribution of this study was to introduce the notion that customer participation in software development requires the selection of one or more customer-developer links through which information can be exchanged. As an initial step toward understanding link selection and use, an exploratory study of 31 software development projects was undertaken to determine the perceived effectiveness of various links and the extent to which they are used in practice.


Three lessons can be drawn based on the results of the study. First, using the link metric for customer participation, we found that more successful projects employed more links than did less successful projects. Hence, the first lesson is that managers should err on the side of providing more links. Second, we observed an abundance of indirect links among many of the projects in our sample. The indirect links were visible in the form of intermediaries and/or customer surrogates that manifested themselves in a variety of different ways. We argue that indirect links are less desirable to use because of information filtering and distortion that can occur. Consequently, our second lesson calls for reduced reliance on indirect links.

Taken together, the first two lessons underscore the notion that developers are best served by establishing numerous direct links through which information can be exchanged between developers and customers to enhance their mutual understanding. This exchange of information cannot take place when the number of links is small or when the channels are distorted by intermediaries.

The third lesson is based on an examination of the differences between the two environments that we studied, custom and packaged software. In each, there was one link that was widely used and highly rated in one environment, but rarely, if ever, used in the other environment. Therefore, in a more speculative vein we offer a third lesson: namely, that development managers would do well to consider using links that have evolved outside of their particular development environment.

Finally, given the intense interest in customer participation in software development from both practitioners and from researchers, we propose that the notion of customer-developer links is an effective framework for both. Practitioners can benefit by stepping back and taking stock of the links that are currently being used in their organizations. These links can be compared against the inventory of links presented earlier to identify gaps and to evaluate excessive reliance on indirect links. Researchers, in turn, can use this approach as a first step toward quantifying the study of user involvement and participation using an objective measure that is common across any development organization.

Acknowledgments.

The authors wish to acknowledge the helpful comments provided by Joey George, Eph McLean, Karen Holtzblatt and the reviewers on earlier drafts of this article. We also want to thank the many development managers who were willing to participate in the research and share their experiences and insights with us. 

References

1. Bostrom, R.P., and Heinen, J.S. MIS problems and failures: A socio-technical perspective—part I: The causes. *MIS Q.* 1, 3 (1977), 17–32.
2. Byrd, T.A., Cossick, K.L., and Zmud, R.W. A synthesis of research on requirements analysis and knowledge acquisition techniques. *MIS Q.* 16, 1 (1992), 117–138.
3. Carmel, E., and Becker, S. A process model for packaged software development. *IEEE Trans. Eng. Manag.* 41, 5 (1995).
4. Churchman, C.W., and Schainblatt, A.H. The researcher and the manager: A dialectic of implementation. *Manag. Sci.* 11, 4 (1965), B69–B87.
5. Daft, R.L., Lengel, R.H., and Trevino, L.K. Message equivocality, media selection, and manager performance: Implications for information systems. *MIS Q.* 11, 3 (1987), 353–366.
6. Grudin, J. Interactive systems: Bridging the gaps between developers and users. *IEEE Comput.* 24 (Apr. 1991), 59–69.
7. Hirschheim, R., and Klein, H.K. Realizing emancipatory principles in information systems development: The case for ETHICS. *MIS Q.* 18, 1 (1994), 83–109.
8. Ives, B., and Olson, M.H. User involvement and MIS success: A review of research. *Manag. Sci.* 30, 5 (1984), 586–603.
9. Kochan, T., Cutcher-Gershenfeld, J., and MacDuffie, J.P. *Employee Participation, Work Redesign and New Technology: Implications for Public Policy in the 1990s*. Commission on Workforce Quality and Labor Market Efficiency, U.S. Department of Labor, May 1989.
10. Lees, J.D. Successful development of small business information systems. *J. Syst. Manag.* 38, 8 (1987), 32–39.
11. Mumford, E., and Henshall, D. *A Participative Approach to Computer Systems Design*. Associated Business Press, London, 1979.
12. Schuler, D., and Namioka, A. *Participatory Design: Principles and Practice*. Erlbaum, Hillsdale, NJ, 1993.
13. *Software Industry 1993 Business Practices Survey*. Price Waterhouse, 1984.
14. von Hellens, L.A. Conditions for Success in the Design and Implementation of Packaged Software: A Study of Accounting Software for Small Companies in the United Kingdom. Ph.D dissertation, Oxford Institute of Information Management, 1990.
15. Von Hippel, E. Lead users: A source of novel product concepts. *Manag. Sci.* 32, 7 (1986), 791–805.
16. Yin, R.K. *Case Study Research: Design and Methods*. Sage, Beverly Hills, CA, 1984.

About the Authors:

MARK KEIL is an assistant professor of Computer Information Systems in the College of Business Administration at Georgia State University. Current research interests include software project management, user involvement, and implementation issues associated with information systems. **Author's Present Address:** CIS Department, Georgia State University, P.O. Box 4015, Atlanta, GA 30302-4015; email: cismmk@gsusgi2.gsu.edu

ERRAN CARMEL is an assistant professor at The American University in Washington D.C. Current research interests include development practices in packaged software organizations and the international competitive implications of these practices. **Author's Present Address:** Kogod College of Business Administration, The American University, Washington D.C. 20016-8044; email: ecarmel@american.edu

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.