INPUT

STRATEGIC MARKET PERSPECTIVE

User Experience with Object Technology

Client Server Program

MARCH 1997

User Experience with Object Technology

INPUT®

Frankfurt • London • New York • Paris • San Francisco • Tokyo • Washington D.C.

INPUT®

About INPUT

Clients make informed decisions more quickly and economically by using INPUT's services. Since 1974, information technology (IT) users and vendors throughout the world have relied on INPUT for data, research, objective analysis and insightful opinions to prepare their plans, market assessments and business directions, particularly in computer software and services.

Contact us today to learn how your company can use INPUT's knowledge and experience to grow and profit in the revolutionary IT world of the approaching millennium.

Subscription Services

- Information Services Markets
  - Worldwide and country data
  - Vertical industry analysis
- Systems Integration / Professional Services
- Client/Server Software
- Outsourcing
- Information Services Vendor Profiles and Analysis
- Internet Opportunities
- Electronic Commerce
- U.S. Federal Government IT Markets
- IT Customer Services Directions (Europe)
- Software Support (Europe)

Service Features

- Research-based reports on trends, etc. (More than 100 in-depth reports per year.)
- Frequent bulletins on events, issues, etc.
- 5-year market forecasts
- Competitive analysis
- Access to experienced consultants
- Immediate answers to questions
- On-site presentations
- Electronic report delivery

Databases

- Software and Services Market Forecasts
- Software and Services Vendors
- U.S. Federal Government
  - Procurement plans (PAR, APR)
  - Market Forecasts
  - Awards (FAIT)

Custom Projects

For Vendors—Analyze:

- Market strategies and tactics
- Product/service opportunities
- Customer satisfaction levels
- Competitive positioning
- Acquisition targets

For Buyers—Evaluate:

- Specific vendor capabilities
- Outsourcing options
- Systems plans
- Peer position

Other Services

- Acquisition/partnering searches

Contact INPUT at: info@input.com, or http://www.input.com

Frankfurt • Perchstatten 16, D-35428, Langgöns, Germany, Tel. +49 (0) 6403 911 420, Fax +49 (0) 6403 911 413

London • Cornwall House, 55-77 High Street, Slough, Berkshire, SL1 1DZ, England, Tel. +44 (0)1753 530444, Fax +44 (0)1753 577311

New York • 400 Frank W. Burr Blvd., Teaneck, NJ 07666, USA, Tel. (201) 801-0050, Fax (201) 801-0441

Paris • 24, avenue du Recteur Poincaré, 75016, Paris, France, Tel. +33 (1) 46 47 65 65, Fax +33 (1) 46 47 69 50

San Francisco • 1881 Landings Drive, Mountain View, CA 94043, USA, Tel. (415) 961-3300, Fax (415) 961-3966

Tokyo • 6F#B, Mitoshiro Bldg., 1-12-12, Uchikanda Chiyoda-ku, Tokyo 101, Japan, Tel. +81 3 3219-5441, Fax +81 3 3219-5443

Washington, D.C. • 1921 Gallows Road, Suite 250, Vienna, VA 22182, USA, Tel. (703) 847-6870, Fax (703) 847-6872

M&S 495/01 1.96

Abstract

This report is written for the executive who is considering, or has just begun, a move to object-oriented (OO) software development and for the vendor who desires to better address clientele concerns in this burgeoning market.

This report discusses user experiences regarding object-oriented application development tools and is based on a survey of 58 vendors, 23 executives, and 37 programmers. It provides a comprehensive review of critical issues associated with a transition to object-oriented programming and how potential obstacles may be overcome.

Object-oriented technology has emerged from its grounding in subroutines to gain more credibility in the business community as a viable alternative to traditional coding practices; it is no longer just a fad. Vendors now need to emphasize the significant advantage that early adopters of this technology will enjoy over the 'wait and see' majority. The marketing challenge to gain wide acceptance of object-oriented technology must address the following issues:

- The organizational, managerial, budgetary, and cultural challenges represented by OO technology, as well as the technology factors, including the resistance of programmers to depart from traditional programming methods
- The need to present a stronger financial case of the benefits of OO technology, which will be supported by changes in management metrics to reflect object creation and reusability
- The requirement to be more sensitive to the hype-fear of IS executives and programmers, as in general, vendor claims far exceed manager and programmer experience

The executive overview provides a summary of the research findings, analysis, conclusions, and recommendations of the report. The following section addresses critical background issues such the current levels of OO usage and reasons offered why OO is not currently employed in certain organizations. The report then presents an analysis of the data related to the benefits claimed by vendors and those expected and actually received by executives and programmers. The final section reports on the present offerings of OO vendors and their relation to the migration of IS department budgets from 1996 to 1998.

The report contains 65 pages and 23 exhibits.

Published by
INPUT
1881 Landings Drive
Mountain View, CA 94043-0848
United States of America

Client/Server Software Program

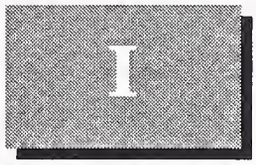
User Experience with Object Technology

CLA6 • 870 • 1997

Table of Contents

Exhibits

V



Introduction

The move from traditional programming methods such as C and COBOL to object-oriented (OO) software development techniques has been long anticipated. Vendors of the technology have promised invaluable improvements in software development in terms of time and money savings, program efficiency and robustness, and the ability to develop distributed systems.

These benefits are mostly realized through the concept of *reusing* an object once it is created and stored in a repository. Other developers may then simply search the repository for the necessary building blocks and avoid the task of programming original code to construct a new application. The concept is easy to imagine, but not so easy to make reality.

To IS executives pursuing shorter development cycles and reduced maintenance costs while maintaining high-quality coding, OO programming appears to be a panacea. However, all is not what the popular press would suggest and reports of actual industry applications are somewhat less than glorious. Companies not already employing OO technology are hesitant to do so before the final outcome is determined in the challenge to establish component integration and interoperability standards.

Regarding integration, this contest is being waged primarily between the Object Management Group's CORBA (Common Object Request Broker Architecture) and Microsoft's COM (Component Object Model).

On the interoperability front, Component Integration (CI) Labs' OpenDoc architecture is pitted against Microsoft's OLE (Object Linking and Embedding).

This report considers the status of OO software development from the standpoint of both vendors and adopters of the technology in business

applications. This report answers such questions as: What are the issues confronting those who have implemented the technology? Where is it being used? Where may it potentially be used? Where are the roadblocks and opportunities? What are OO's greatest attributes and failings?

A
Definitions

For the purposes of this report, the following working definitions were used:

- *Object-oriented (OO) technology*

  Includes *Products* such as languages, CASE tools, application developing environments, databases, class libraries and frameworks, and programming utilities, as well as *Services* such as consulting, software development, training and migration

- *Objects*

  Includes programming objects, components, data bundles, and related technologies

Without significant exception, all executives, programmers, and vendors interviewed agreed with these definitions.

B
Objectives

This report had the following major objectives:

- Help marketing managers in software, systems, and professional services firms understand the market for object-oriented technology
- Describe the level of usage of OO technology in certain business functions and its potential use in others areas
- Discuss benefits perceived as opposed to those actually received and hindrances to widespread adoption of OO technology in industry
- Illustrate the gaps between the vendor and user communities
- Present a detailed account of important issues affecting OO technology development dynamics

This report is written for the executive who is considering, or has just begun, a move to OO software development and for the vendor who desires to better address clientele concerns in this burgeoning market.

C
Scope

This report provides executives with strategic insights into the benefits to be gained through OO technology and how to realize them optimally. The report also presents guidelines for implementation and suggestions for smooth adoption. Though not a tutorial, this report is intended to present a conceptual framework to help management consider object-oriented technology as a tool for success in the 1990s and beyond.

The research focused on organizations located in the U.S.; however, the same findings may also be relevant in other developed countries. Extensive interviews were conducted with OO vendors and with developers, managers, and executives who are current or potential users of OO technology.

The vendors included in the report are prominently recognized providers of OO products and/or services. Current possessors or likely purchasers of the technology that were surveyed were companies with at least $25 million in 1995 revenue.

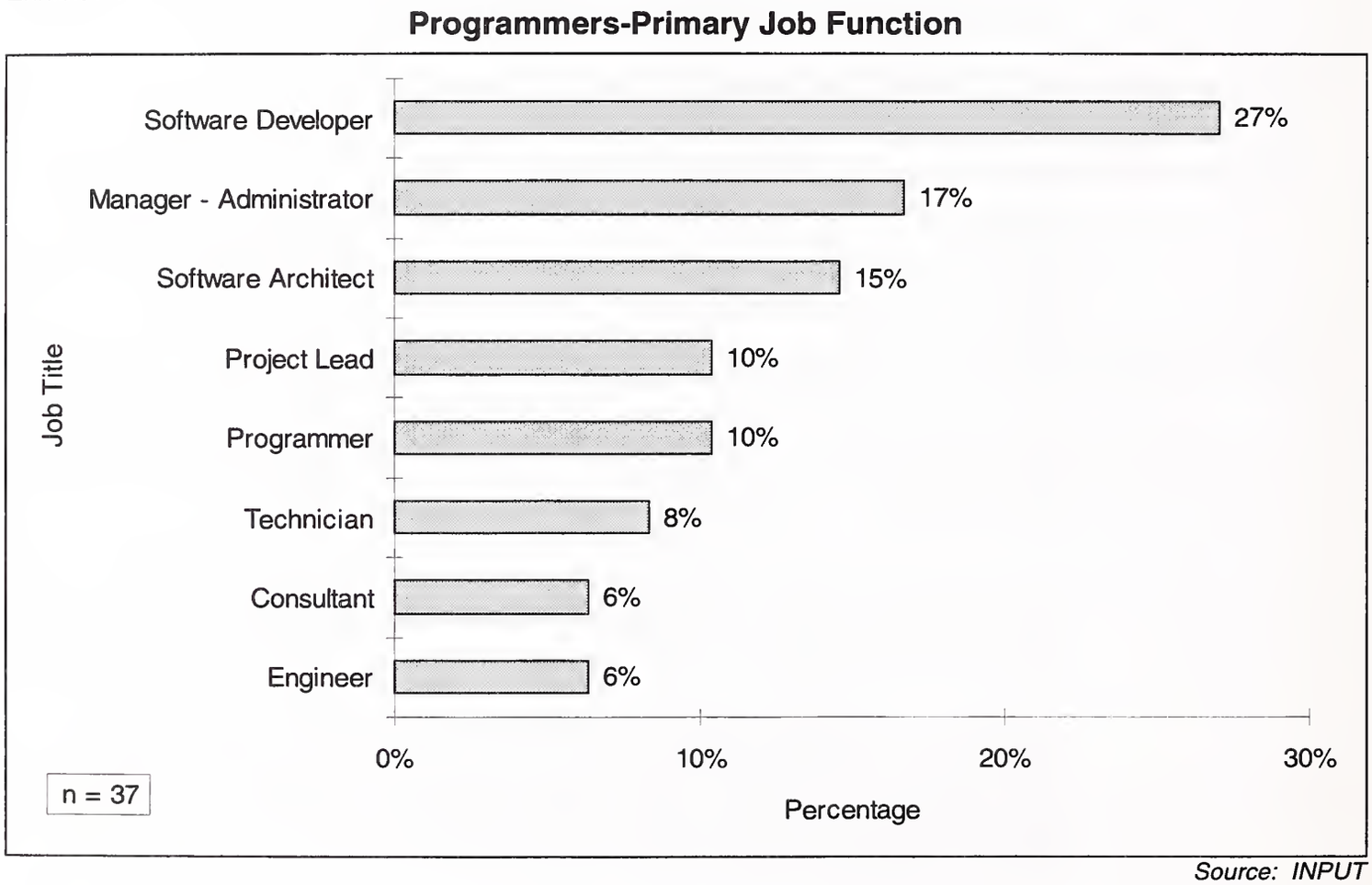
Data was also collected regarding large educational institutions and government purchasers. All major industry sectors are covered by the report, as well as all organization sizes where significant IS activities are present.

The timescale addressed is late 1996 to 2000. Given the rapid, ongoing development of OO technology, greater emphasis is given to the near term of early to mid-1997. Changes are occurring almost daily in this actively evolving industry.

D
Methodology

The research is based on interviews with fifty-eight vendors, thirty-seven programmers, and twenty-three IS executives, which are analyzed in detail. Reviews of published materials, on-line resources, and case studies were also conducted to compile this report. The breakdown of those classified as "programmers" is presented in Exhibit I-1.

Exhibit I-1



Source: INPUT

E
Report Structure

The following is a brief description of the organization of this report.

- Chapter II is an Executive Overview providing a summary of the research findings, analysis, conclusions, and recommendations of the report.

- Chapter III, The Object-Oriented Technology Landscape, addresses critical background issues such as the current levels of OO usage and reasons why OO is not currently employed in certain organizations.
- Chapter IV, Benefits of Object-Oriented Technology, presents an analysis of the data related to the benefits claimed by vendors against those expected and actually received by executives and programmers.
- Chapter V, Object-Oriented Products and Services, reports on the present offerings of OO vendors and their relation to the shift in IS department budgets from 1996 to 1998.
- Appendix A provides definitions specific to OO technology.
- Appendix B lists leading client/server and standards organizations.
- Appendix C provides the three questionnaires used in gathering data for this report.

F
Related Reports

Related reports in INPUT's Client/Server Software Program include:

- *Component Software Battles: ORBs, OLE, and OpenDoc*
- *Client / Server Systems Management Software*
- *Middleware: Is DCE the Answer?*
- *Object-Oriented Platforms for Client / Server Systems*
- *Client / Server Explosion-How Users Choose Platforms*
- *Worldwide Client / Server Market Forecast, 1995 - 2000*

In addition, INPUT reviews vendor strategies in its Vendor Analysis Program (VAP). Vendors of object-oriented technology that have been reviewed in the VAP include:

One Wave, (Business@Web, Inc.)

NeXT Software, Inc.

Open Market, Inc.

Microsoft Corporation

(Blank)

II

Executive Overview

This section provides a summary of the major conclusions and recommendations made in this report concerning current user experience with the practical use of object technology.

A
Stressing Early Adoption Benefits

Object-oriented (OO) technology has emerged from its grounding in subroutines to have increasing credibility in the business community as a viable alternative to traditional coding practices; it is no longer just a fad.

There is now need to emphasize the significant advantages that early adopters of object-oriented technology will gain over the 'wait and see' laggards.

To gain wider acceptance of object-oriented technology, the following issues must be addressed (Exhibit II-1):

- Business environment realities—These include the organizational, managerial, budgetary, and cultural challenges represented by OO technology, as well as the technology factors, including the resistance of programmers to depart from traditional programming methods

Exhibit II-1

OO Technology Marketing Challenges

- Promote change in business culture
- Present strong financial arguments
- Support realizable advantages

- The need for a solid financial case—The need to present a stronger financial case of the benefits of OO technology that will be supported by changes in management metrics to reflect object creation and reusability

- Avoidance of overselling benefits—The requirement to be more sensitive to the resistance of IS executives and programmers to overhyped marketing messages, as in general, vendor claims far exceed manager and programmer experience

B

Promoting Change in the Business Environment

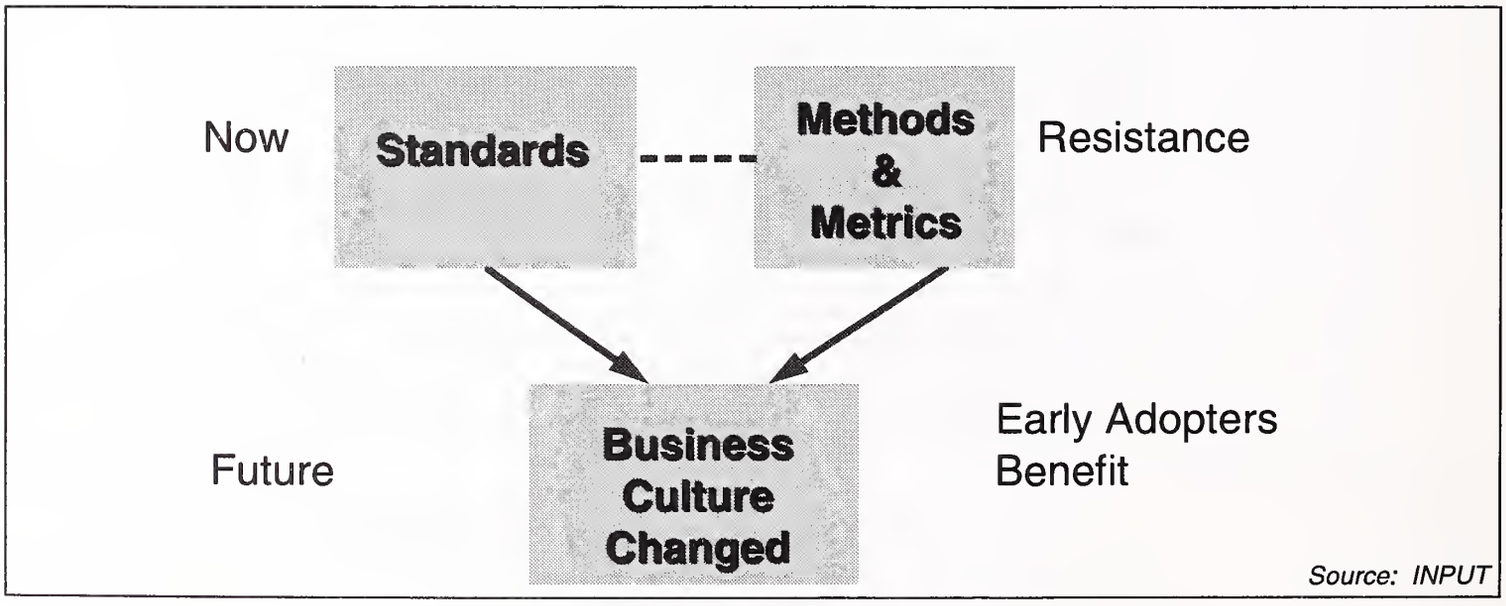
Resistance to early adoption of OO technology is encountered on the technological, organizational, managerial, budgetary, and cultural fronts.

Technologically, the move to OO is a big step and the dominant hurdle from which the majority of other obstacles are generated.

There will be significant benefits earned by organizations that are prepared to accept the available standards and work with them to gain organizational and technological experience (Exhibit II-2).

Exhibit II-2

Integration of Object Technology and Business Organization



Three major issues that are technology related are addressed below:

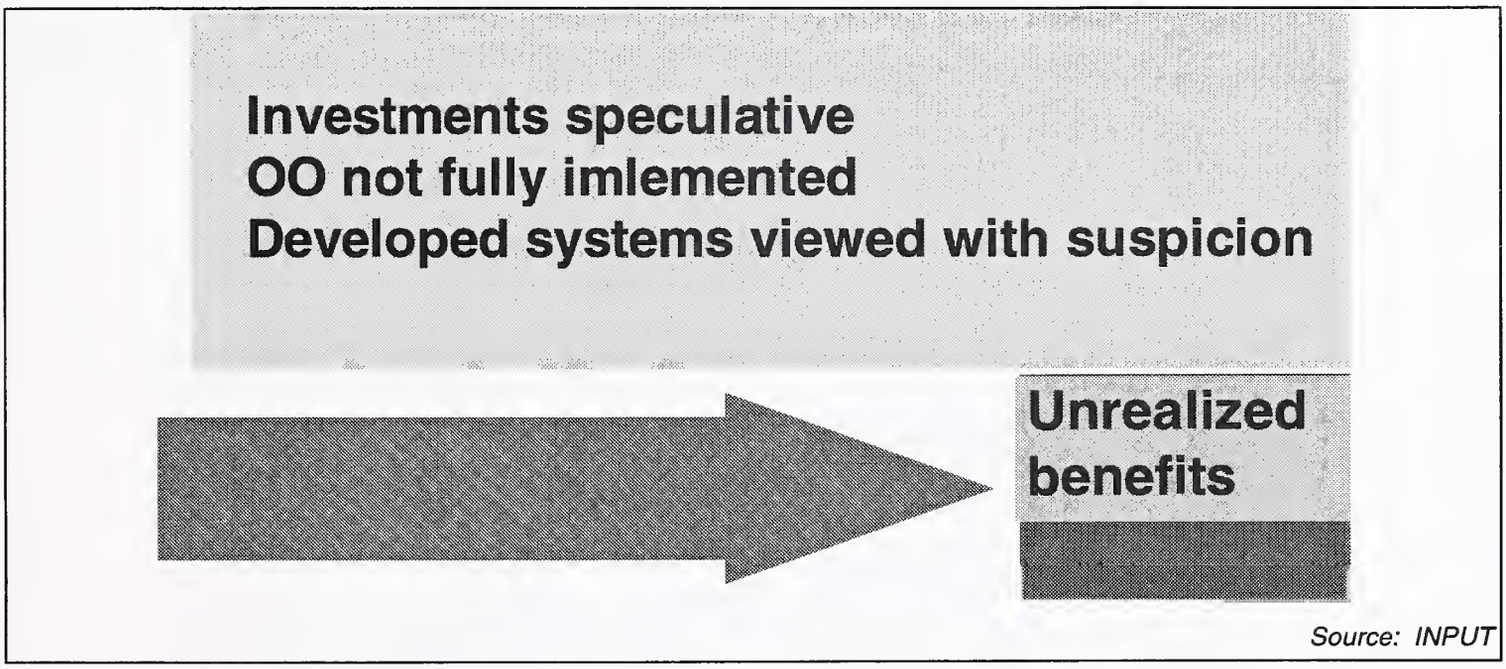
- Low acceptance of the available standards for interoperability and integration of object-oriented systems
- The need for changes in the way programmers are managed and given incentive
- The need for stronger implementation methodologies

1. Low Acceptance of Available Standards

The issue of interoperability and integration standards is a significant challenge to the object-oriented technology business environment (Exhibit II-3).

Exhibit II-3

Interoperability and Integration Standards Issues



Companies are reluctant to make the commitment to a particular set of OO tools prior to the concrete establishment of accepted protocols that prescribe how objects are created and how they interact.

The contest between the Open Group and Microsoft to establish dominant standards is still undecided.

Meanwhile, prospective OO adopters are largely content to make do with current technology or are wary of the significant up-front expenditure and perceive the investment as speculative.

The establishment of integration and interoperability standards must be expedient or all parties lose. Vendors stand to lose the most, as customers are reluctant to procure OO technology in the absence of standards.

Programmers are frustrated as they endeavor to link inconsistent OO platforms via ad-hoc middleware.

Consequently, executives are disappointed when expected benefits from OO technology are not realized either, because it is not implemented at all or the robustness of the resultant piecemeal systems is compromised.

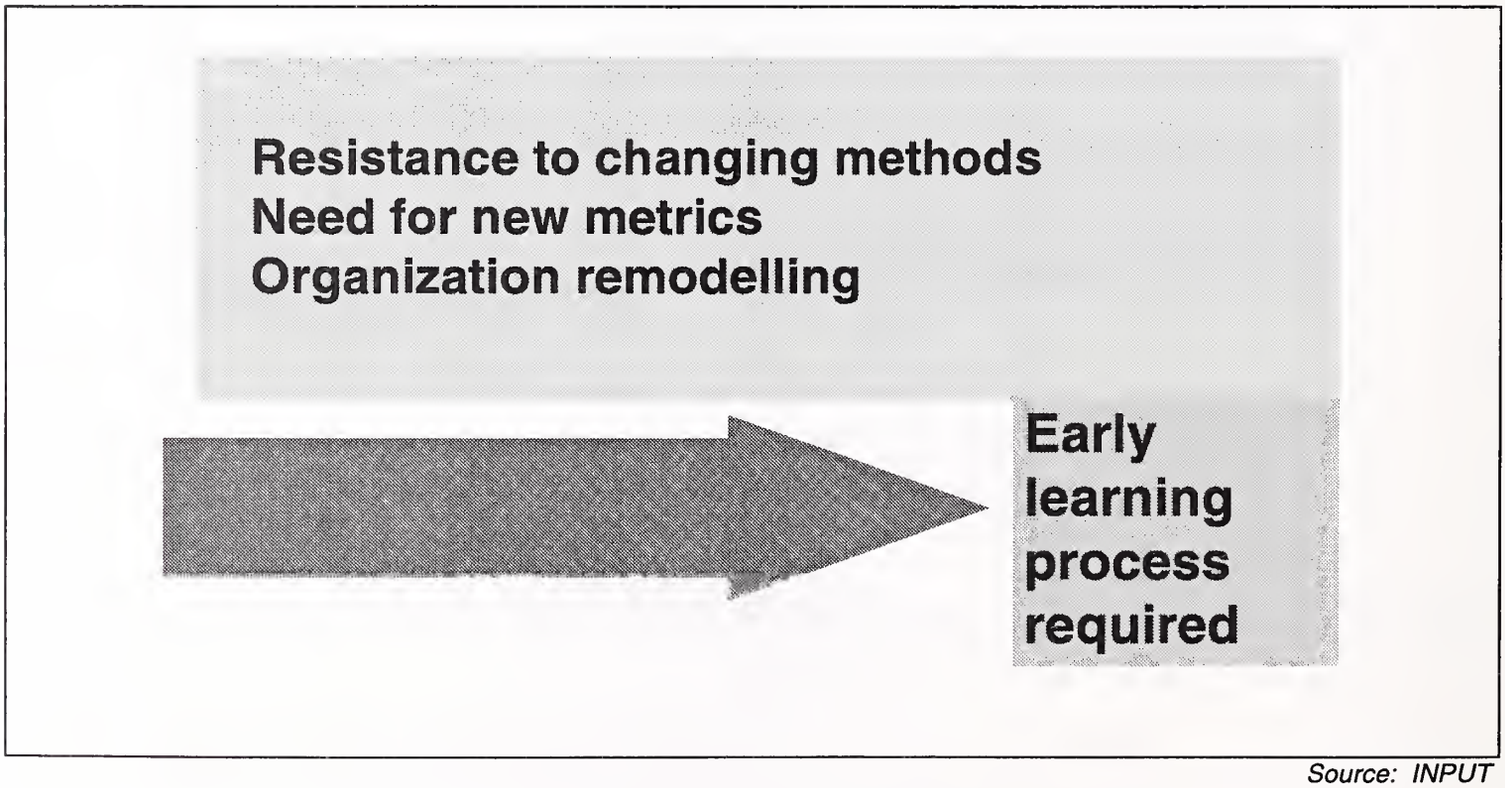
Early adopters will gain a significant advantage over those with a 'wait and see' attitude toward OO technology standards.

2. Need for Programmer Attitude Change and Incentives

Creating the reusable blocks of code requires a significant shift in programming methodology and indeed, a shift in a programmer's perspective of the organization's business functions (Exhibit II-4).

Exhibit II-4

Programmer Environment Issues



Source: INPUT

Companies are wary of a move to object orientation due to the organizational changes required.

A move to OO isn't as simple as just training programmers in a new language; they must learn to develop applications in a radically different manner and become familiarized with the concept of reuse.

To accomplish this, customary metrics of programmer performance must be modified dramatically to effect a smooth transition and maximize object creation and reuse.

The sooner these changes are enacted, the sooner a firm can not only realize the immediate benefits associated with OO technology, but can implement the enterprise-wide vision required to effect the transition.

Code reuse requires reusable classes-OO software components that developers assemble to build applications.

However, developing these classes is not simple and it can require much more time to develop reusable object components than it does to develop conventional, nonreusable software.

Reusable component development requires:

- A more detailed and consistent set of analysis, design, and programming skills
- A different set of objectives-maximum reuse instead of fast development
- Much tighter quality assurance and testing

Object orientation divides a programming task into sufficiently small parts that underlying patterns become apparent.

In a large system, the same sort of tasks may occur at different levels. Once these similarities are recognized, a design approach and much of the code that performed one task may be reused for another.

When personnel at all levels of an organization identify the congruencies, better reusability, more formalized thought processes, and aesthetic consistency result.

Programmers are generally resistant to change from the "traditional" original code development process.

The shift from creating original code to developing and reusing objects is the largest obstacle to the acceptance of OO technology.

Selling the idea to programmers will be the greatest challenge in organizations where the use of languages such as C and COBOL is deeply embedded in the application development culture.

Achieving reuse requires not only a rich library of components that have been designed specifically for reuse, but application developers who can locate the needed reusable components and are given incentives to use them once they are identified.

To effect a complete transition to OO technology throughout an enterprise, programmers must no longer be evaluated against conventional metrics such as the number of lines of code produced. Rather, their performance should be assessed according to the number of objects created and the amount of code reused.

Programmers must learn to adopt an enterprise-wide view to enable the construction of globally applicable objects and ensure their optimal reuse.

The human organization should be remodeled in a manner that corresponds to the new OO technology. Objects are small entities that collaborate closely with their peers using formalized communication methods. The developers behind their creation should function in the same manner.

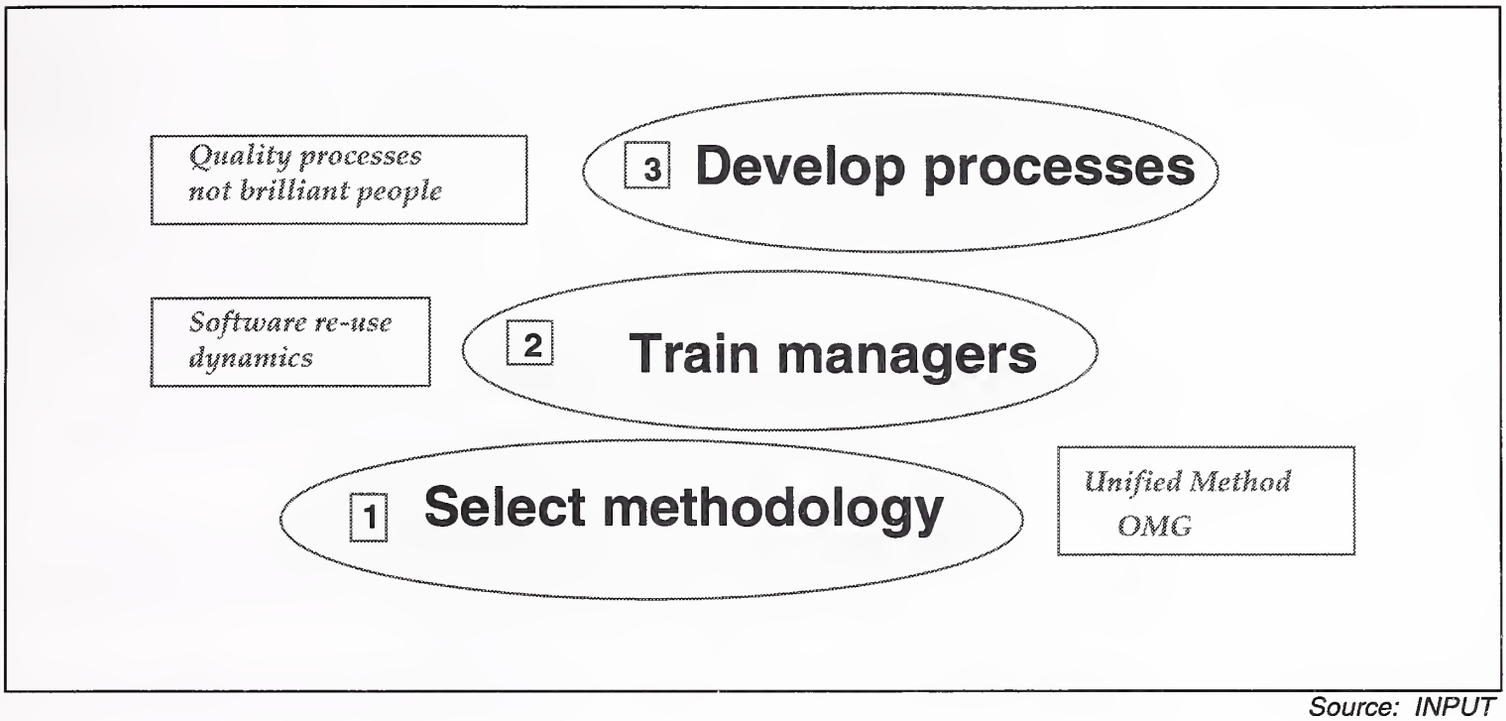
3. Implementation Methodologies

In addition to the battle for standards regarding component integration and interoperability, the challenge to establish a development methodology standard has entered the foray.

Success in using OO implementation methodologies will, however, not only be a function of the standard used, but also of the management environment and the process quality achieved (Exhibit II-5).

Exhibit II-5

Implementing Object Technology Systems



Source: INPUT

However, one of the biggest problems facing IS managers moving to OO application development is selecting a methodology.

There are presently 15 to 25 methodologies on the market, each offering its own application development system.

The leading approach is the Unified Method, which combines several independent methodologies and has received the endorsement of companies with large OO projects.

By virtue of support from the developer and corporate applications communities, there is a high probability that the Unified Method will indeed become the de facto standard.

The Object Management Group (OMG) is also diligently preparing a standard modeling language aimed at enabling vendor compatibility by improving interoperability between object analysis and design tools.

Once a methodology for reuse has been ascertained and developers have been properly trained, attention then focuses on the selection of tools best suited to promote reuse.

Organizations with little or no experience devising formal development strategies should employ an outside consultant with a proven track record or hire a person who has worked with a good methodology.

Simple, fundamental changes in the way application development is managed, combined with careful selection of tools and programmer training, can significantly increase the amount and quality of reusable code.

Ongoing training is another hidden cost of implementing formal reuse. Only through comprehensive, hands-on training of developers will an OO-adopting organization achieve reuse rates that will begin to pay back the investment.

Developers should look beyond the simple creation of a library of reusable components.

Having an object repository-a multiuser database of reusable frameworks and objects-allows developers to take advantage of new modules and enhanced application features as they're added, without reprogramming existing applications.

When programming teams are building applications, they must attempt to maintain a fine granularity for object modules without resorting to an arbitrary metric such as the number of lines of code.

The finer the granularity, the more reusable the resulting components will be.

A reuse-oriented discipline begins with the analysis phase preceding the actual reuse of classes and frames. A formal methodology that defines how objects are developed documents module names, properties, and methods. It also delineates interfaces and establishes frameworks for reuse.

Implementing a reuse methodology will make IS engineers more efficient. Many developers are familiar with reuse, yet report struggling to implement it effectively.

Organizations that have successfully adopted OO technology and achieved improved development efficiency through reuse have managers who understand the dynamics of software reuse as well as their technical people do, and organize the development effort accordingly. They do not, however, have more brilliant designers or more skillful developers than anyone else.

A fundamental change is presaged: the shift from pre-industrial development approaches to those appropriate to the post-industrial world of components, assemblies, and services.

Creativity is required in the process of software development, not in the creation of the software itself; it is a quality issue.

Although the proper tools and methodology are required to implement reuse, absolute success is dependent upon the quality of the personnel involved and their commitment to the project.

Reuse should be championed by the team leaders and IS management members who are responsible for authoring, formalizing, and documenting reuse methods.

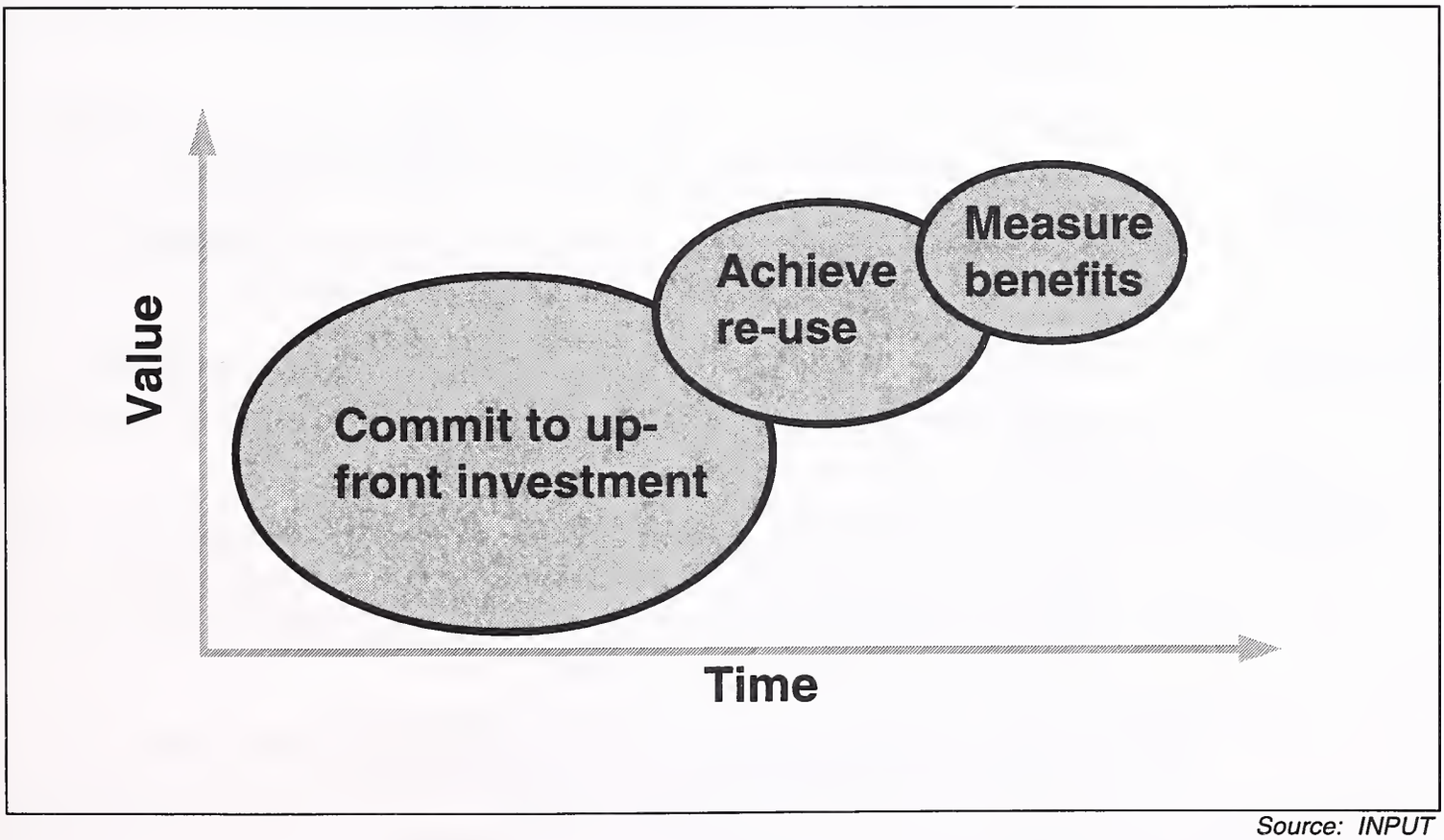
C

Presenting Strong Financial Arguments

Object technology will begin to gather momentum in the marketplace as the financial imperative becomes established. Currently there exist a number of obstacles to the development of a clear financial argument for the adoption of object technology (Exhibit II-6).

Exhibit II-6

Object Technology Value Development



Source: INPUT

There is a lack of significant financial evidence as yet to support argument in favor of a shift to object orientation.

Financial benefit claims are loosely couched in terms of improved software quality, robustness, and time savings through reuse.

However, object-oriented technology is too immature for a large number of documented cases of cost savings to exist.

The principal benefits of OO programming-increased productivity and decreased maintenance-depend on reuse.

If substantial reuse isn't practiced, the company will have switched from conventional programming to OO development without fully realizing the benefits.

Managers must be able to measure the results achieved; if it can't be measured, it can't be managed. The simplest way to measure reuse is to track how many lines of original code are required to bring a new project to fruition.

Ideally, as reuse increases, developers will be able to reap the productivity benefits and managers will see a decrease in freshly written code.

Managers must fight the fear of up-front expense and time lag before realizing the true benefits of OO technology through reuse.

Vendors must work to overcome these fears by striving to develop sales contracts based on ultimate value realized by the customer rather than on a flat fee.

Hard evidence of financial reward, not just academic and vendor hearsay, is required to effect the widespread adoption of OO technology. Time and money savings benefits must be supported by improved code quality and reduced program maintenance, as well.

Creating reusable elements can take two to three times longer than programming using conventional languages. It is this up-front delay and expense that has many OO-adopting organizations frustrated as they await payback on their investment.

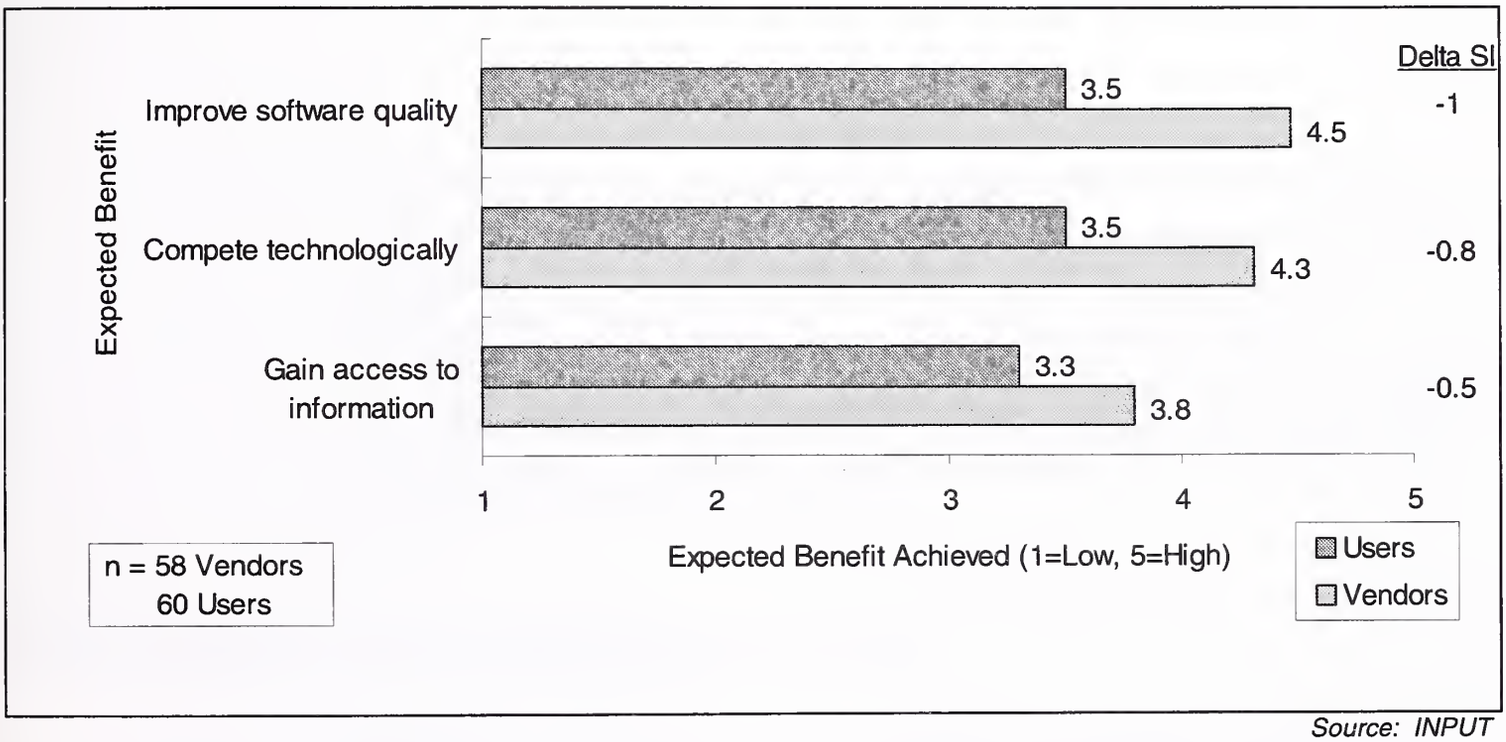
D

Avoiding Technology Oversell

Exhibit II-7 depicts the difference between vendor, executive, and programmer opinions regarding the applicability of object-oriented tools to assorted business functions.

Exhibit II-7

Targeted versus Received Benefits of OO Technology



Source: INPUT

The three categories shown-improved software quality, competitive positioning through technology, and access to information-received the highest rankings of the ten categories respondents were asked to evaluate.

In general, vendor claims far exceeded executive and programmer experience.

Vendors must be careful not to oversell their product. Managers and programmers are wary of the hype and will not buy in easily.

To convince managers, it is not necessary to win over their subordinates, though, to ensure a successful business relationship, an OO-adopting organization's programmers should be a vendor's strongest ally in selling the concept to management.

Vendors of OO technology should work to bridge the knowledge gap between executives and programmers regarding OO technology capabilities. As these opinions merge, a stronger demand for OO products and services will arise.

In the absence of vendor conformance with respect to object integration and interoperability, potentially adopting organizations are hesitant to "take the plunge."

To alleviate this problem, vendors should actively participate in and promote the adoption of standards.

III

The Object-Oriented Technology Landscape

This chapter discusses the significant differences between the responses of the three types of interviewees. Recommendations are made for narrowing the gaps between the groups regarding claims, perceptions, and actual experiences with the technology so as to achieve greater customer satisfaction levels. Finally, it forecasts the evolution of the technology: how recent and forthcoming developments will affect the widespread adoption of object-oriented (OO) technology, the software development community, and the performance of business functions.

A
Object-Oriented Technology Experience Levels

Exhibit III-1 illustrates the levels of experience reported by vendors of OO technology, as well as those of programmers and executives of adopting organizations.

On average, vendors of OO technology purport 6.5 years of experience-over twice the average amount of experience reported by programmers (3 years) and over three times that of executives (2 years).

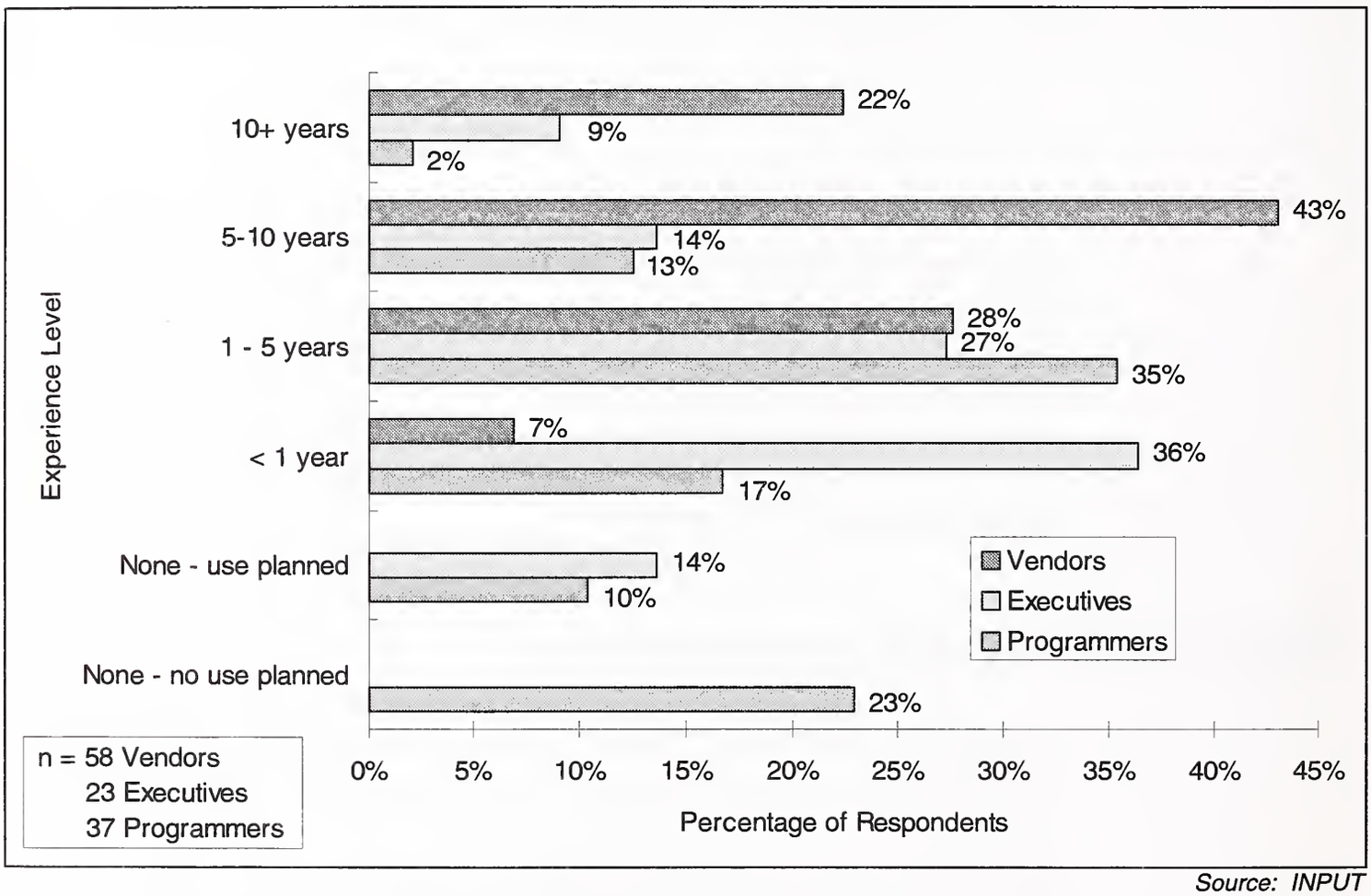
The largest proportion of vendors (43%) reported an experience level in the range of five to ten years, programmers (35%) one to five years, and executives (36%) less than one year.

These differences help explain the disparity between expectations and application potential responses among the unique ranks. The vendors have the greatest amount of experience in applying OO technology and are thus inherently enthusiastic about its applicability and usefulness.

Programmers are perhaps the hardest sell since they'll be the ones actually working with the tools. Accordingly, they consistently exhibit the lowest expectations and levels of satisfaction with OO use. This may be attributed to their reluctance to abandon the familiar C and COBOL programming languages in lieu of the new paradigm.

Exhibit III-1

Experience with OO Technology



Source: INPUT

The shift from traditional programming methodologies to OO requires the software developer to envision the application in the context of the entire organization; to adopt an enterprise-wide perspective.

This enables the coder to create objects with an increased likelihood of being reused to create new programs across unique business functions, thus decreasing the amount of original code required.

B

Why OO Technology Is Not Currently Employed

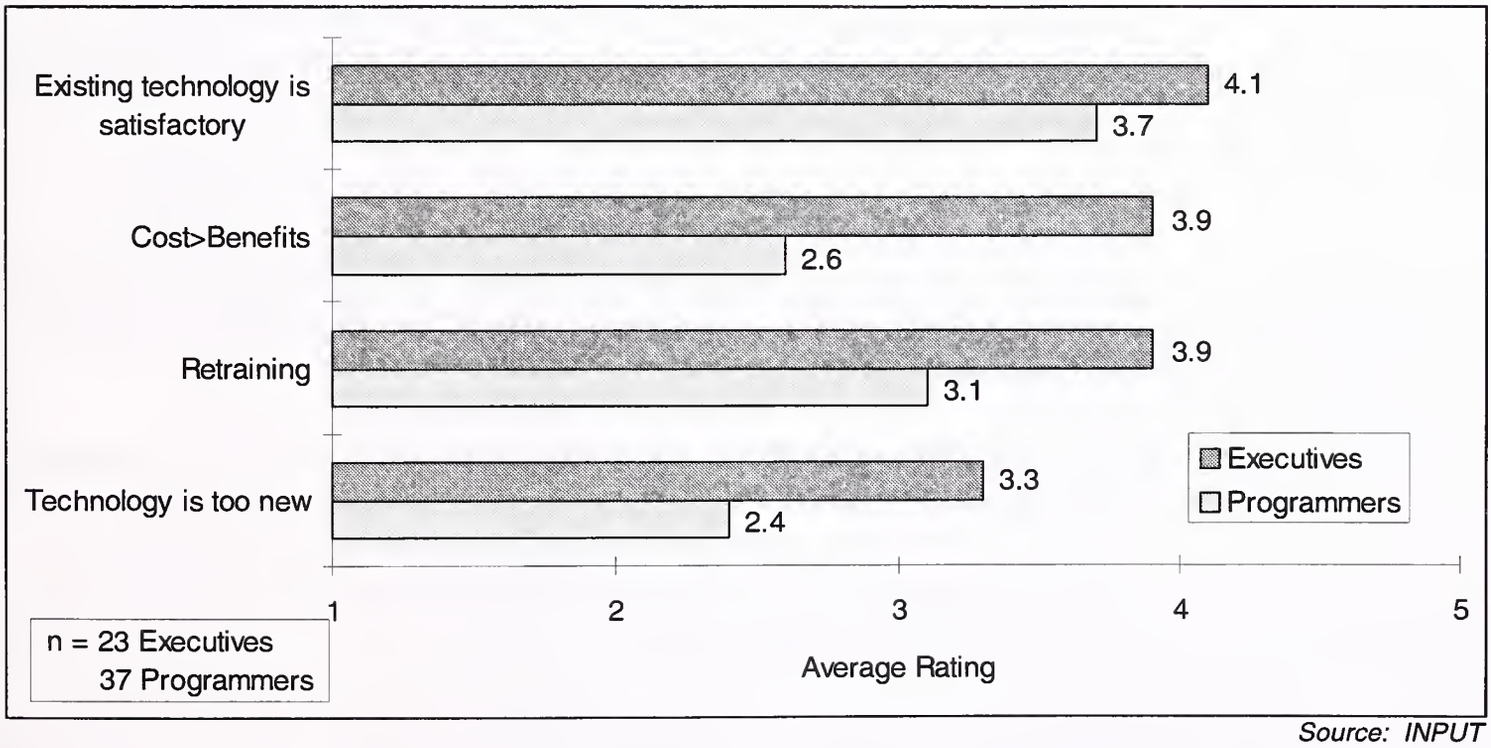
1. Existing Technology

A breakdown of the responses given by executives and programmers about why OO technology is not currently used is given in Exhibit III-2. The reason cited most often by both executives and programmers for not presently employing OO technology is that existing technology is satisfactory. This can, however, imply several factors lending to this response.

Though respondents may truly believe that existing means are sufficient and that no significant reason exists for the enterprise to adopt a "new" technology, it is likely that this response is grounded in failure to recognize the benefits of OO platforms and fear of opportunity costs. This is not an indictment of the executives' or programmers' ignorance in this respect, but rather support for the lack of hard evidence linking application of OO technology to the purported benefits.

Exhibit III-2

Reasons Why OO Technology Is Not Currently Used



Source: INPUT

This scenario is not specific to OO technology, but influences many strategic IS investment decisions. Perhaps the most widely touted benefit of OO technology use is the ability to reuse coded objects stored in repositories.

These objects may be linked to create unique programs regardless of their ultimate application. For example, an object employed in a logistics optimization program may also lend itself to an inventory control application.

The objects are "generic" insofar as they function in the same manner irrespective of input; only data format standards must be honored-their internal manipulation of data is consistent.

2. Cost

The benefits associated with such programming methodology include the obligatory "save time and money" through improved coding efficiency. A programmer may create an object with multiple functionality, thereby saving time creating other applications. Programmers must no longer "reinvent the wheel" to create an original program. They may simply select and link the appropriate object "building blocks" from the repository to create a new application. However, "simply select" is an exception to the rule at this point in the evolution of OO applications development.

For an organization to realize the benefits obtainable through OO technology fully, the objects *must* be reused. However, objects designed with reuse in mind take an estimated two to three times longer to develop. Object orientation advocates concur that paying the dues on the first OO project will have significant long-term payoffs.

While supporting cases are well documented, this initial step is still presenting a significant hurdle to organizations contemplating OO application. Though overhead associated with reusable objects was once cost prohibitive, innovations in storage and processing in recent years have all but eliminated this concern.

Reuse of objects offers organizations the opportunity to cut costs associated with software purchase, development, and maintenance. Furthermore, by focusing on objects rather than applications, it is easier for designers to adopt an enterprise view and identify classes and relationships that cut across functions and organizations.

Object-oriented technology offers many benefits to an employing organization. However, as with a large number of IS-oriented projects, it also commands a significant initial investment with uncertain outcome. Developing application objects and establishing the repositories in which to house them often require a prohibitive amount of time and capital.

Once created, however, OO applications can offer many additional benefits, such as improved software quality, information access, operating efficiency, and technologically competitive positioning.

3. Training

The necessary expense to train a staff of programmers properly in object-oriented languages can be considerable. Managers may be unwilling simply to forget the costs already incurred to train these same programmers in traditional languages, initially.

Indeed, why not scrap the whole lot and hire programmers fresh from college who already know how to program in objects and can be hired at a fraction of the wage of a conventional veteran programmer? This is not a practical option for several reasons, not the least of which would be the pervasive crushing of the organization's morale!

Another compelling argument against this scenario is that programming objects to suit multiple business functions requires application developers to adopt an enterprise-wide view. In doing so, these objects may be applicable in a variety of functions; hence, their reuse is maximized-a significant if not primary objective of a migration to object orientation.

The programming veterans of the enterprise will be an invaluable resource for providing this comprehensive vision of subtle interdepartmental relationships, thus improving the likelihood of widespread object reuse.

4. Object-Oriented Technology Is Immature

Though object-oriented technology is a fairly common topic in IS circles, its successful application in industry is still largely unproven. That it works is not the issue; rather, are the claimed benefits truly tangible and of the magnitude expected? In this respect, there is a lack of hard evidence that is preventing a large number of companies from taking the plunge and becoming early adopters.

Those that do take a bold step forward, however, will have the advantage in the future. A move to object orientation requires more than just learning how to program in a new language; it requires the formal rethinking of applications development to envision and create reusable objects. The sooner this culture is fostered, the sooner the embodying organization will begin to realize the benefits attainable through OO programming.

C
Potential Uses of OO

Object-oriented technology and application development has been applied in virtually every business function imaginable. For the purposes of this study, ten primary sectors were selected for OO applicability and three personnel classifications surveyed.

The results of this survey are represented by the respective respondent class in Exhibits III-3 through III-5. Vendors predictably believe strongly that OO technology will affect virtually every category, whereas executives are less enthusiastic, and programmers even less than executives.

For the chosen business functions however, there is general uniformity of expectations regarding the applicability of OO technology. A composite of the four highest average rated functions is presented in Exhibit III-6.

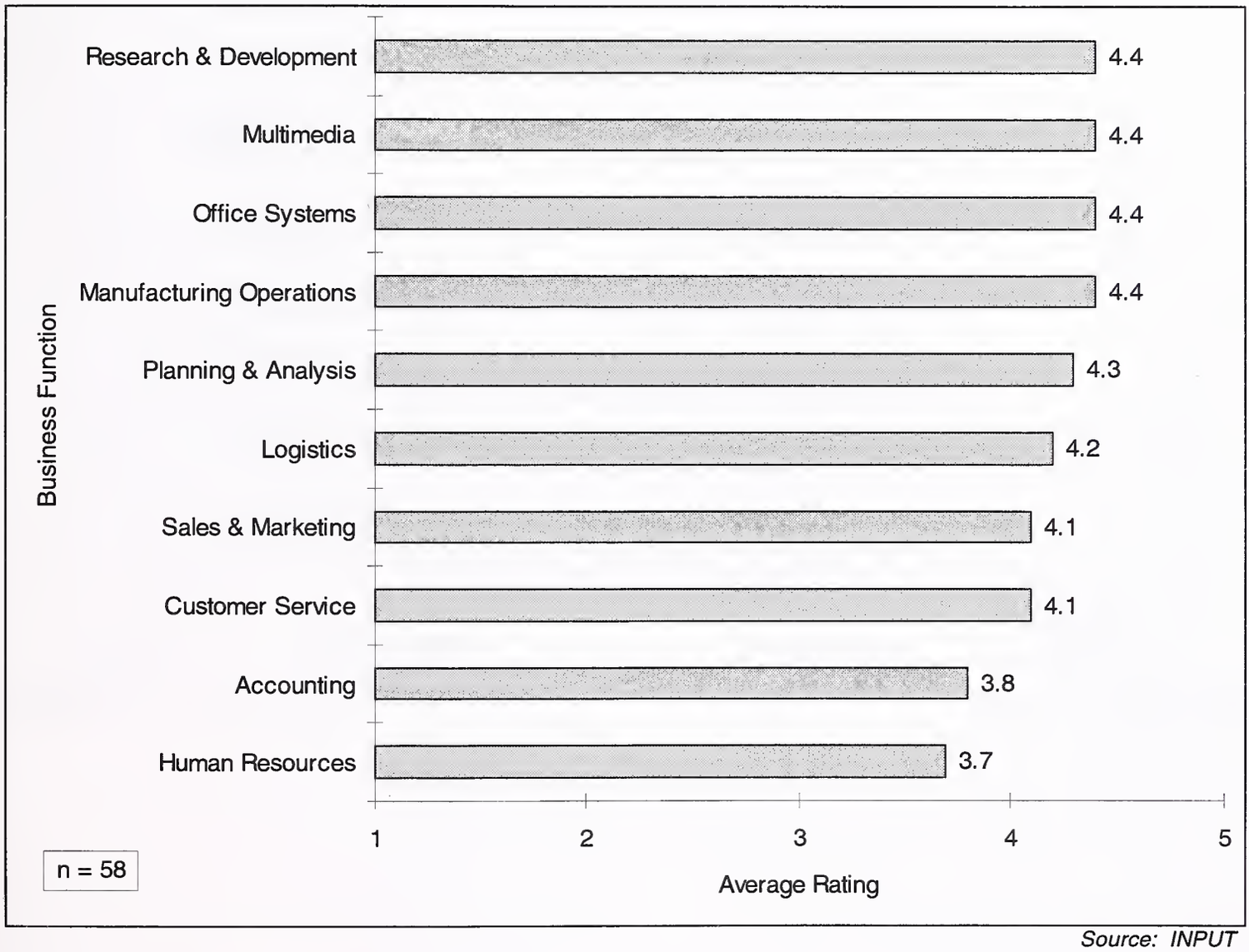
Observations that can be made from the data on the following charts include:

- Executives awarded the highest potential ratings to the four categories of Sales and Marketing, Planning and Analysis, Research and Development, and Multimedia.
- Programmers were less convinced of such potential and rated only the functions of Customer Service and Sales and Marketing with average values of 3.9 or above.
- Vendors were predictably the most optimistic in their assertions of OO applicability. Of the ten chosen business functions, only the categories of Human Resources and Accounting earned vendor ratings averaging less than 3.9.
- The category of Sales and Marketing holds the distinction of garnering a common rating greater than 3.9 from vendors, executives, and programmers alike, whereas the Accounting and Human Resources functions received average ratings of less than 3.9 overall.
- The categories of Manufacturing Operations and Logistics show high promise to vendors, but executives and programmers weren't quite convinced and gave the respective functions only marginal ratings. Vendors must adjust marketing messages to address this discrepancy.

- Averaging vendor, executive, and programmer ratings, the four business functions with the highest potential for OO application are Sales and Marketing, Planning and Analysis, Research and Development, and Multimedia. Each of these functions received an overall average rating of 4.0 or greater.

Exhibit III-3

Vendors-Potential OO Use by Business Function



Source: INPUT

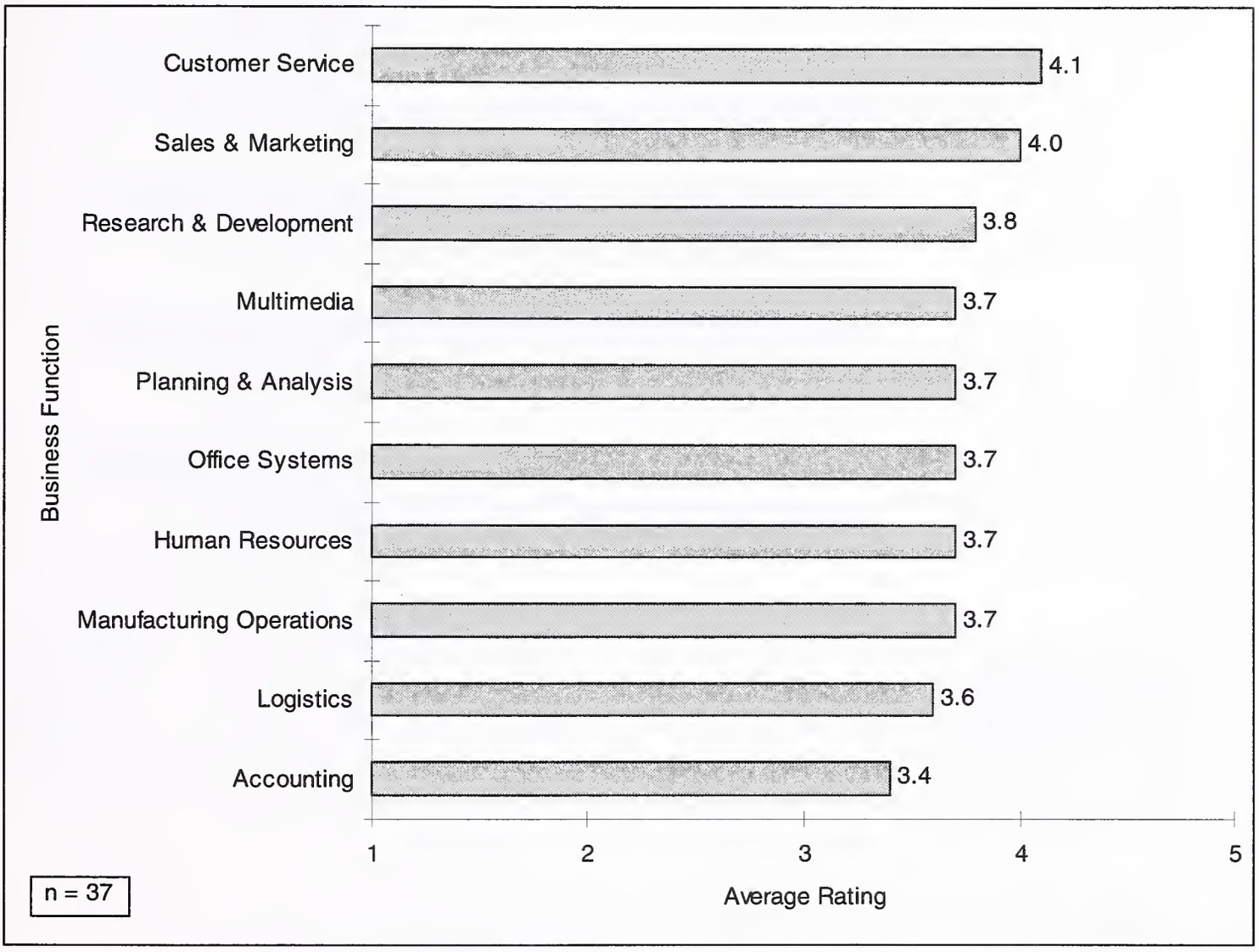
Exhibit III-4

Executives-Potential OO Use by Business Function

Source: INPUT

Exhibit III-5

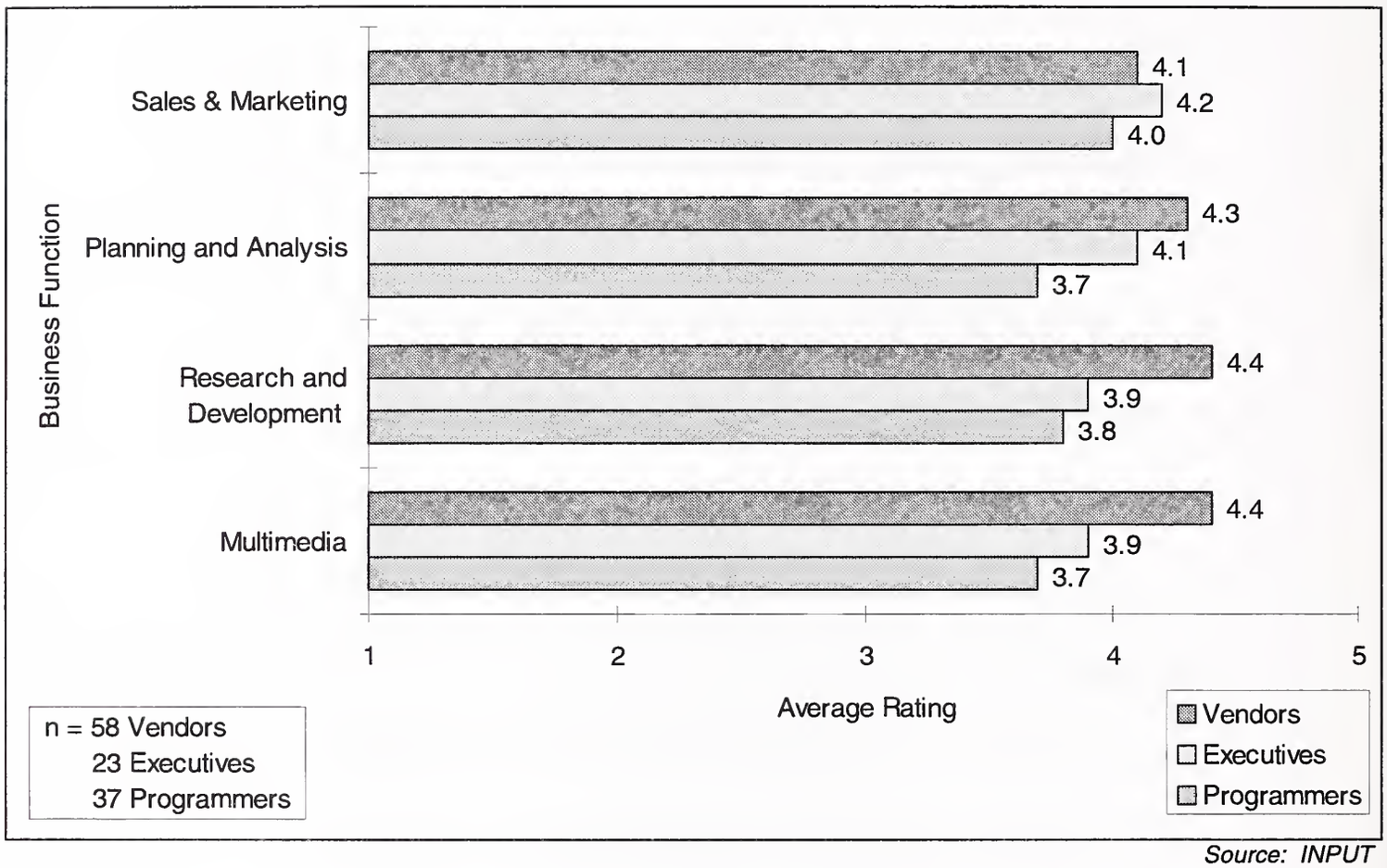
Programmers-Potential OO Use by Business Function



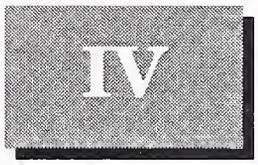
Source: INPUT

Exhibit III-6

Top Functions by OO Application Potential



Source: INPUT



Benefits of Object-Oriented Technology

The introduction of a new technology is always shrouded in a certain degree of mystery regarding its true benefits. A chasm exists between the abilities touted by its creators and proponents and those experienced in "real world" business applications. The use of object-oriented programming languages is no exception.

As with most information technology-related products and services, OO programming is exceedingly technical in nature, thus implying that a "truer" understanding of its subtle nuances is more likely to be encountered at the developer level. This principle appears to hold in the case of OO; programmers are typically the most conservative regarding expectations for the business applicability of OO as well as benefits actually received.

Programmers are the most difficult to convince for another reason. Object-oriented programming represents, in most cases, a radical departure from traditional languages, such as COBOL and C, and requires a fundamental redirection of programming methodology toward the goal of creating and using reusable objects. It is the reuse of preprogrammed modules that enables new applications to be developed by simply assembling the necessary blocks of code.

Respondents were asked to report expected and received benefits from the use of OO technology as manifested in time and money savings, improved software quality, access to information, operational efficiency, and ability to improve their competitive positioning through the use of technology. Vendors were also asked to relate their insights regarding the benefits attainable through the use of OO technology.

A
Targeted versus Expected and Received OO Technology Benefits

Results of the expected versus received OO technology benefits questionnaire are illustrated graphically in Exhibits IV-1 through IV-6 on the following pages.

Of the aforementioned categories, vendors most often cited the virtue of improved software quality. All benefit categories were given predictably high vendor ratings of 3.9 or greater, with the nominal exception of improved information access.

Executives expected the most significant improvements to occur in the areas of time savings, ability to compete technologically, and access to information.

Though all programmer expectations for OO benefits fell below 3.9, they shared somewhat in vendors' enthusiasm for software quality improvement potential and assigned their highest average rating of 3.7 to that category.

Overall, vendor promises were exaggerated by over half a point to those benefits actually received by executives and by over a full point to those received by programmers. Perhaps the most egregious violation occurred in the area of time savings; this was one of the benefits most highly touted by vendors, yet one of the lowest ranked in terms of benefits actually received by executives and programmers.

INPUT believes that this disparity is due primarily to the focus on short-term returns by these parties and not on long-term time savings through reuse. Initially, there may be a significant time lag because programming the objects themselves can take two to three times longer than programming in conventional languages. However, over the long run, programs may be constructed much more quickly by simply assembling the preprogrammed objects.

The category of improved software quality was the only one in which any appreciable conformity existed between the benefits expected and received by both executives and programmers. This was also the category most highly ranked by vendors.

Among executives, expected benefits always exceeded those actually received. The most significant benefits were expected to be in the area of

time savings. However, this category demonstrated the largest deficiency compared to what was actually received.

Again, this difference may be attributed to the time lag between adopting OO programming (i.e., engaging in the laborious and time-consuming exercise of creating a repository of reusable objects) and actually constructing new applications from OO program modules. Nominal differences existed in the remaining categories, with the most agreement occurring with respect to money savings (lowest expected, lowest received).

Likewise, programmers' expectations were never met. Programmers' responses were fairly uniform in the discrepancy between expected and received benefits ratings. On average, actual received benefits were rated a half point lower than expectations, with the highest ratings (both expected and received) awarded to improved software quality (3.7 and 3.4, respectively).

B

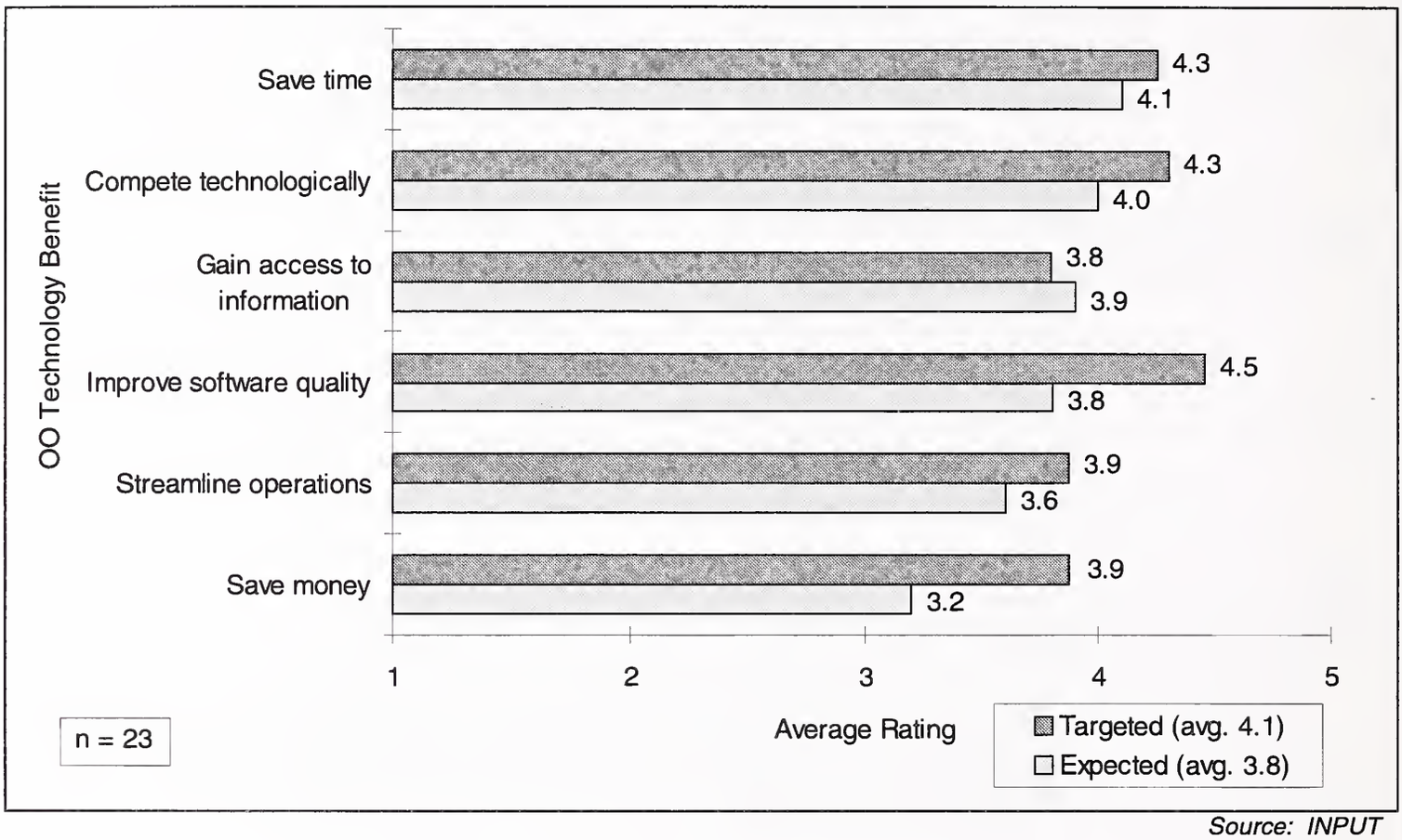
Executives: Vendor-Targeted versus Expected Benefits

From Exhibit IV-1, vendor messages regarding benefits from OO technology concentrate in the areas of improved software quality, gaining competitive advantage through technology, and reducing the time required to develop new software applications.

A significant discrepancy exists, however, in the category of improved software quality between the benefit levels targeted by vendors and those expected by executives. To address this matter, marketing personnel should work to improve their message and increase their emphasis on this facet.

Exhibit IV-1

Executives
Vendor-Targeted versus Expected Benefits



Source: INPUT

C

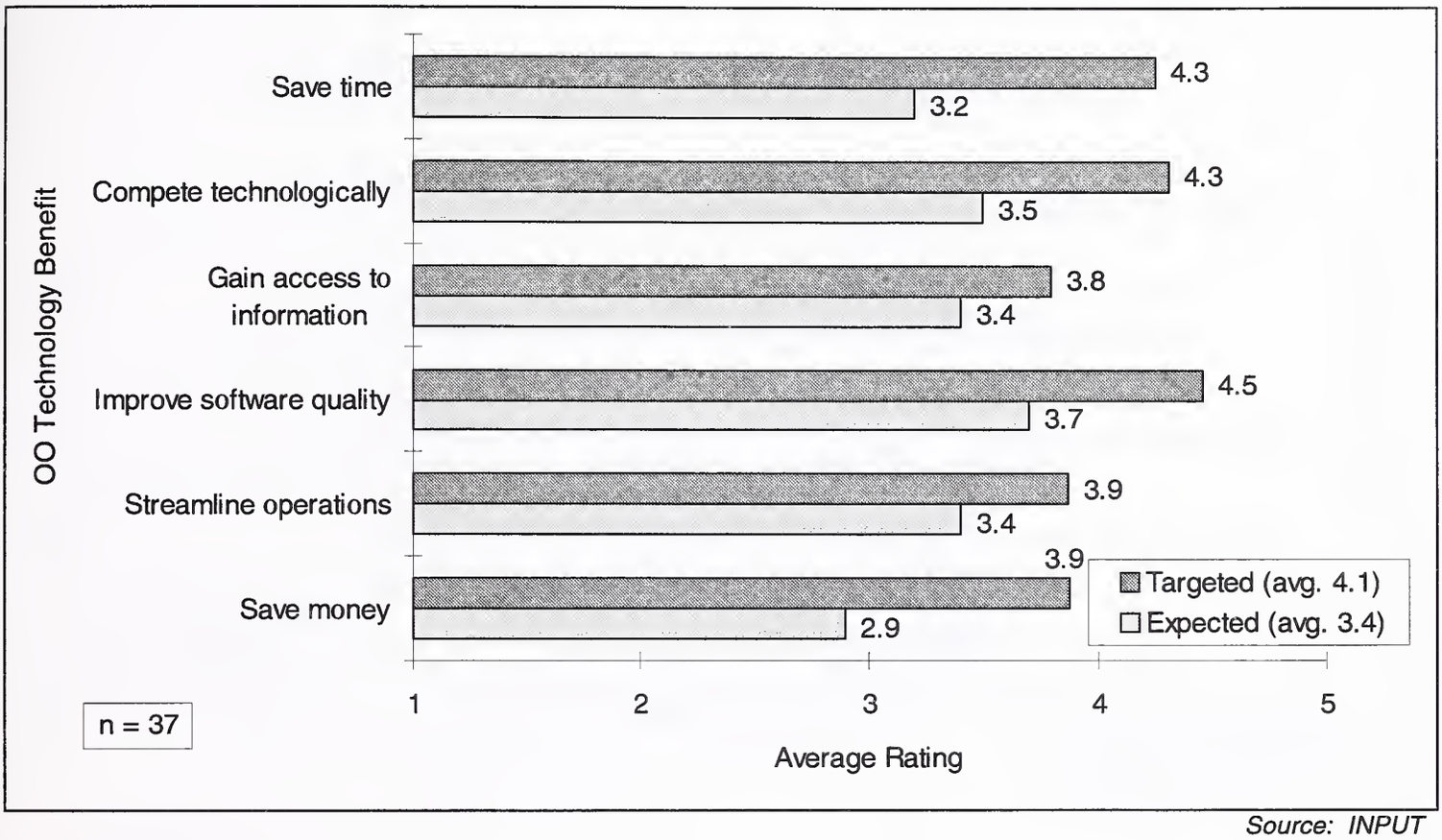
Programmers: Vendor-Targeted versus Expected Benefits

For programmers, the vendor-targeted benefit levels for each of the six categories fell far short of their expectations. As illustrated in Exhibit IV-2, of the three highest categories ranked by vendors, programmer expectation lagged by an average of nearly one point on a 1-5 rating scale. The highest expectations by programmers occurred in the area of improved software quality. INPUT believes that this is due to the technical nature of the category, which programmers may be better able to appreciate.

Across the board, OO vendor messages should be revised to improve their impression on programmers. Messages to programmers should be geared toward first convincing them of the benefits to be gained through object-oriented programming and then leveraging their endorsement to win over executives. Programmers should thus be targeted as the champions of an organization's transition to object orientation.

Exhibit IV-2

Programmers
Vendor-Targeted versus Expected Benefits



Source: INPUT

D

Executives versus Programmers: Expected Benefits

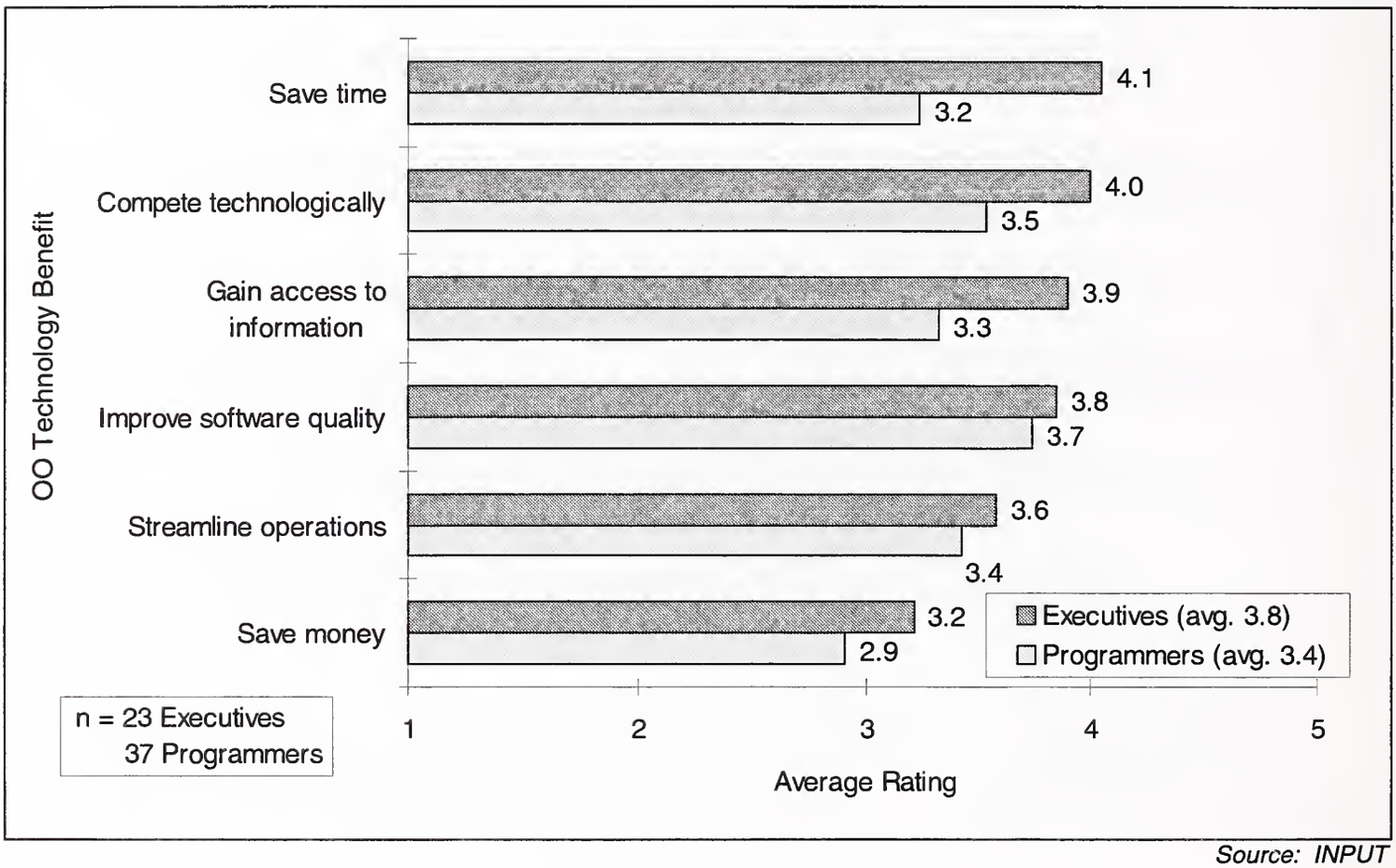
From Exhibit IV-3, executive expectations exceeded those of programmers in all categories, with the most notable discrepancy appearing in the category of time savings. INPUT research suggests that the low ranking allotted to this category by programmers is due primarily to the greater amount of time required to program an actual object.

Programmers presently do not consider the substantially reduced time required to develop future applications by reusing preprogrammed objects an advantage.

Though it is well documented that programming an individual object may take up to three times as long as programming in conventional languages, INPUT believes that a substantial portion of programmers surveyed are still focusing on the short-term time lag while these objects are developed and not on the long-term time-saving benefits realized through their reuse.

Exhibit IV-3

Executives versus Programmers
Expected Benefits



Source: INPUT

E

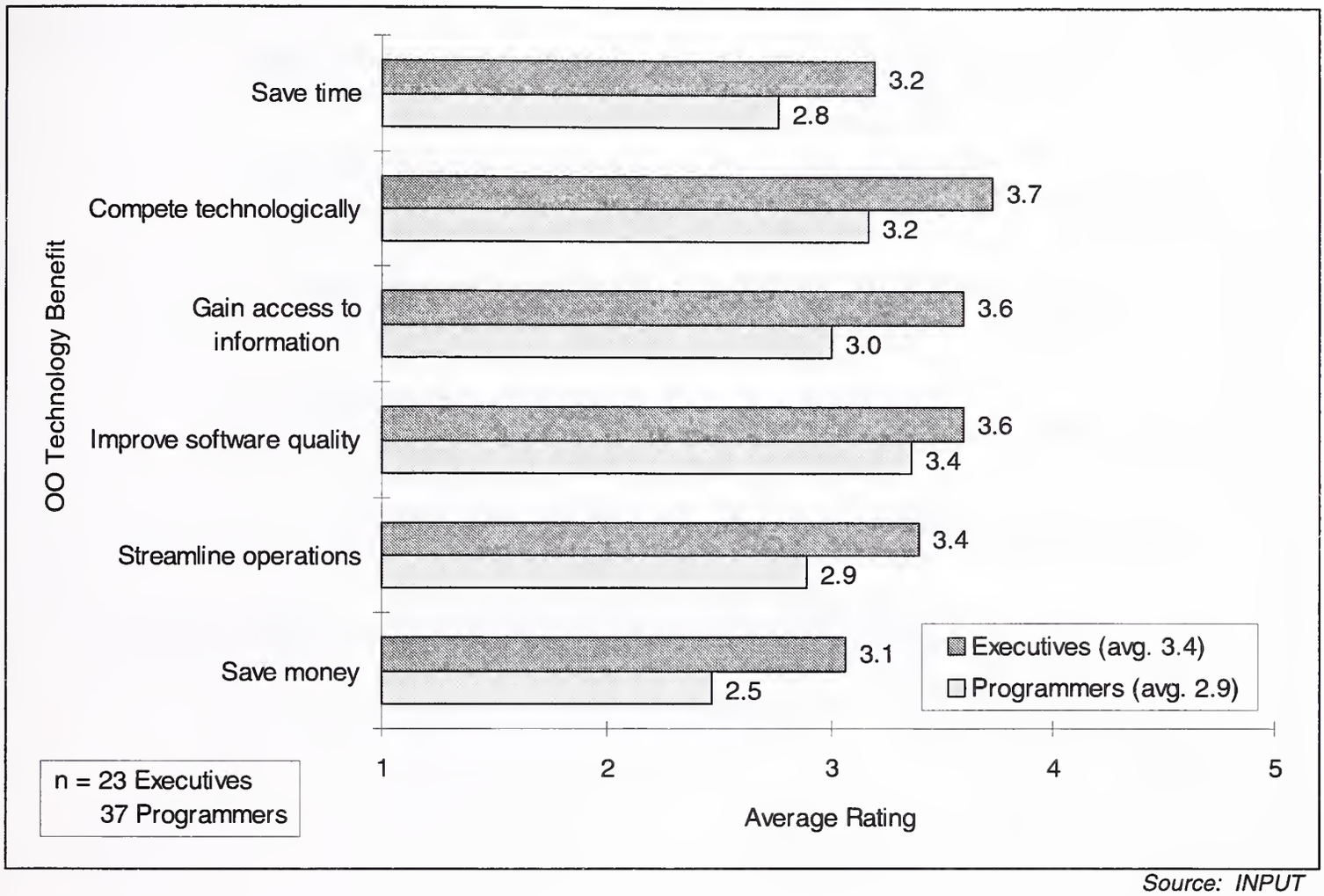
Executives versus Programmers: Received Benefits

As shown in Exhibit IV-4, executives report having received greater benefit from object-oriented technology than do programmers. Programmer experience fell below executive experience by an average of half a point on a 1-5 rating scale.

More importantly, perhaps, the chart also implies a significant difference between the benefits actually received by programmers and the levels executives *believe* their programmers are receiving.

Exhibit IV-4

Executives versus Programmers
Received Benefits



Source: INPUT

F

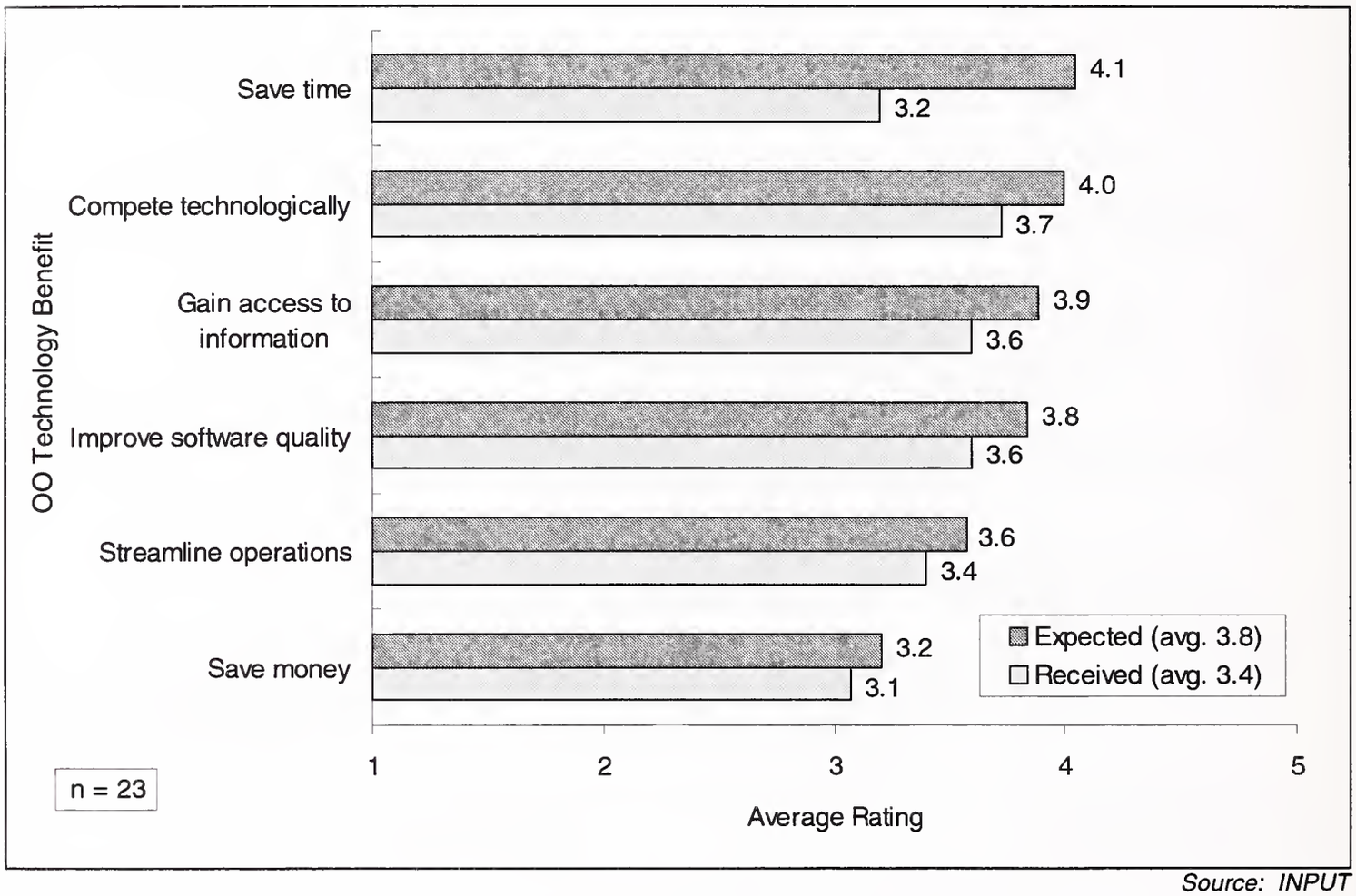
Executives: Expected versus Received Benefits

Exhibit IV-5 shows a comparison of the benefit levels expected by executives and the levels actually experienced. In all categories, expectations were not met; the most significant discrepancy occurred in the area of time savings.

Again, INPUT's research indicates that this is due to the time lag experienced before achieving significant benefits through object reuse. Furthermore, this lag is exacerbated by the lack of proper incentives given to programmers, whose performance is still largely measured against conventional metrics (such as the number of lines of original code produced).

Exhibit IV-5

Executives
Expected versus Received Benefits



Source: INPUT

G

Programmers: Expected versus Received Benefits

As demonstrated in Exhibit IV-6, programmers predominantly expressed low levels of satisfaction regarding benefits obtained through OO. Their expectations were conservative and, as with executives, consistently went unmet.

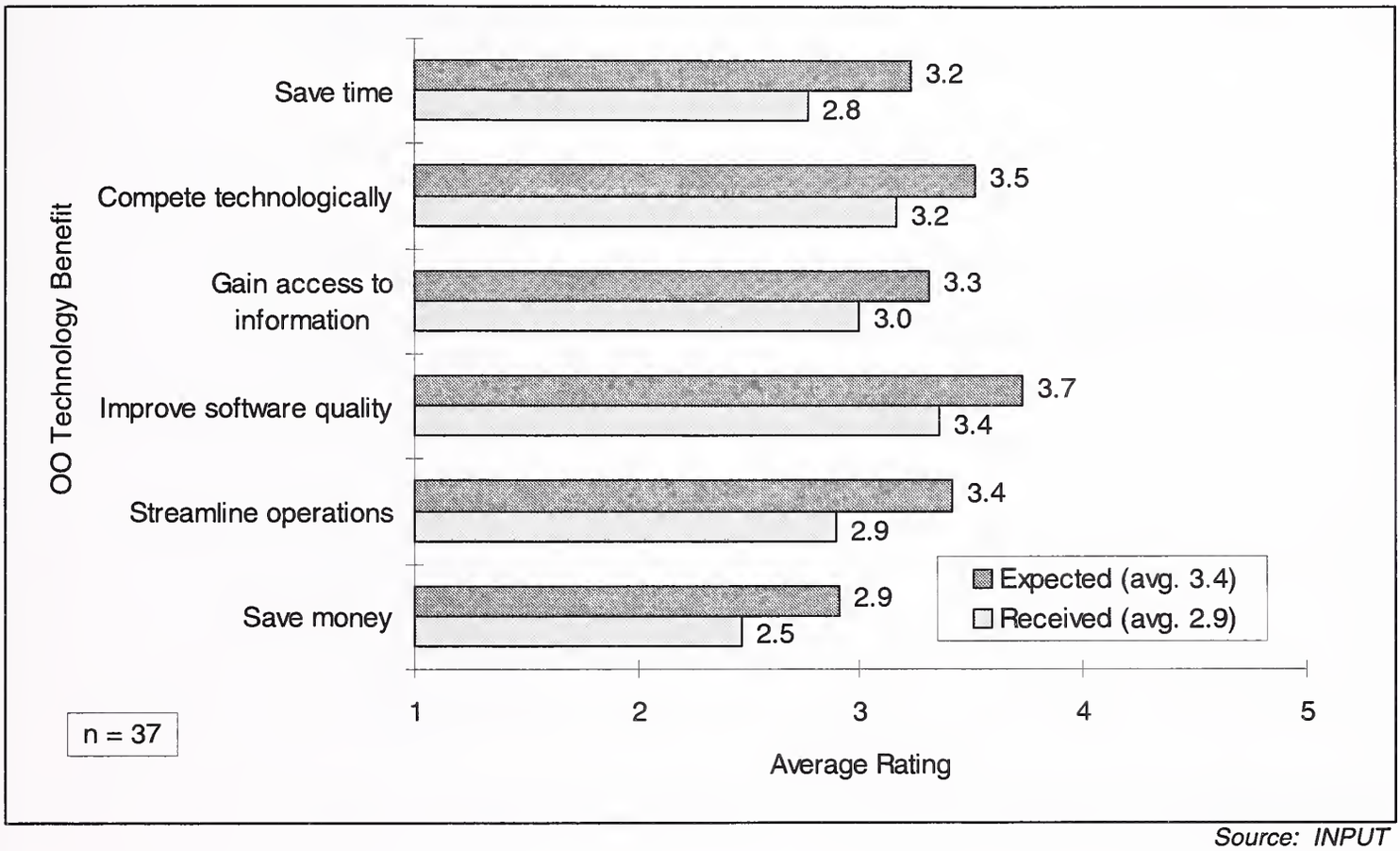
Programmers' highest accolades went to the category of improved software quality, which INPUT believes is attributable to the primarily technical perspective of programmers. Again, their lowest levels of satisfaction occurred in the areas of time and money savings.

In all cases, vendor claims exceeded programmer experience ratings by at least 1 on a scale of 1-5. Vendors must target programmers as the champions

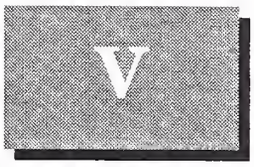
of an organization's transition to object orientation. To accomplish this, vendors should improve mechanisms for customer feedback and include training in their services portfolio.

Exhibit IV-6

Programmers
Expected versus Received Benefits



Source: INPUT



Object-Oriented Products and Services

The genre of OO technology encompasses a wide variety of product and service offerings. For the purposes of this study, the category of OO products was restricted to the following:

- CASE tools
- Visual designs
- Language
- Databases
- Middleware
- Object request brokers (ORBs)
- Class libraries
- Objects

Object-oriented services included in the survey were in the categories of:

- Training
- Consulting
- Software development
- Migration, Internet related
- Project management

A Need for OO Products

1. Executives

Of company executives surveyed, 52% expressed a need for products from at least one object-oriented product category. Of those, 75% demonstrated a desire to purchase OO products from four or more product classifications.

Only three categories commanded a response from half or more of the executives. Of note, however, is the strong alliance between programmers and executives in their expressed need for visual design tools.

The highest percentage of both programmers and executives responded to this category with nearly matching percentages. This high degree of correlation from both management and technical perspectives indicates a true need for products of this nature and a more definitive market.

2. Programmers

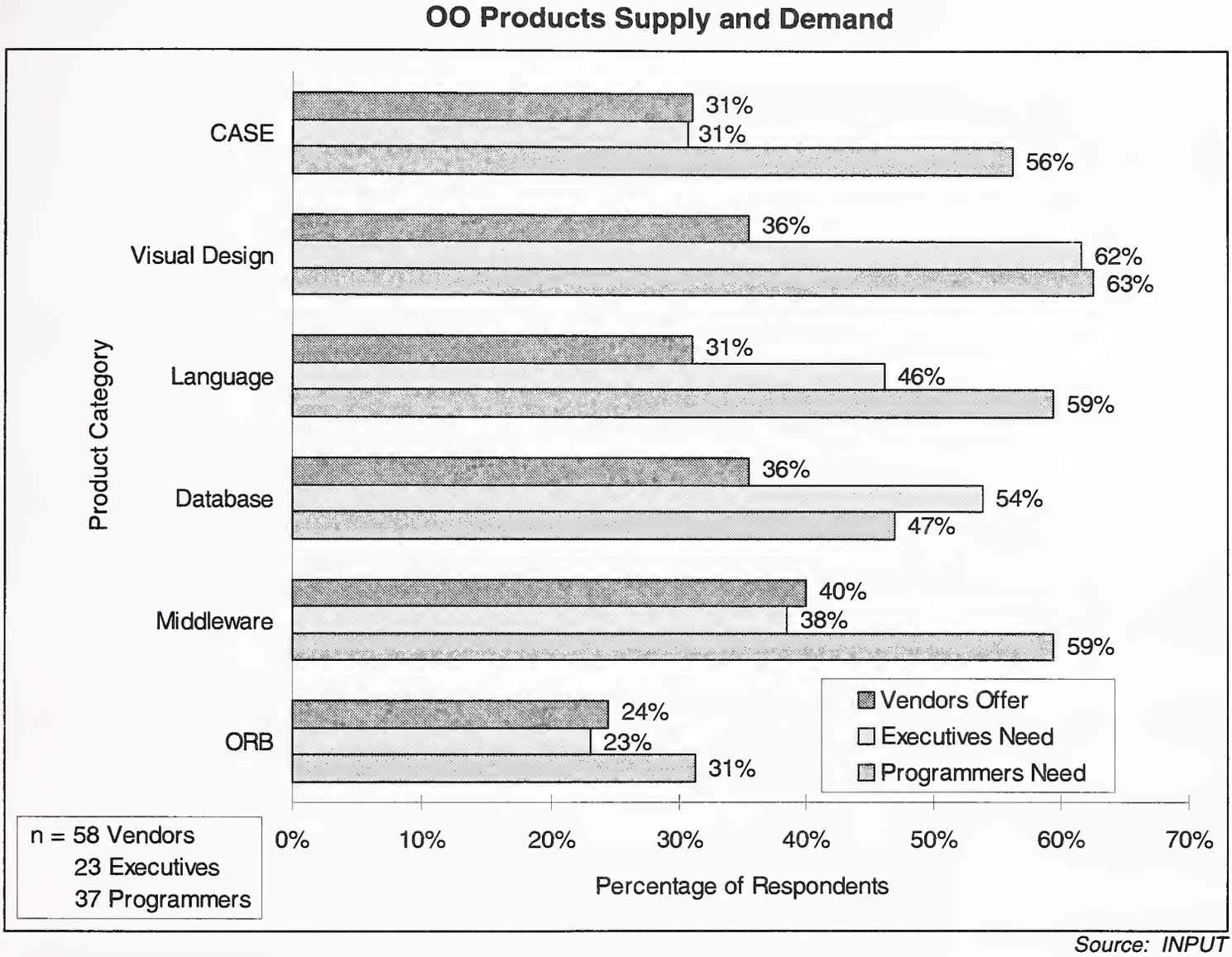
Eighty-six percent of the programmers surveyed expressed a need for products from at least one OO category. Of those, nearly half (47%) demonstrated a desire to purchase OO products from five or more classifications.

In contrast to executives, half or more of the programmers expressed a need for five of the six product categories, with the only true exception being that of ORBs.

Indeed, demand for ORBs solicited the lowest percentage response from both executives and programmers. Again, relative to demand for the other product categories, this correlation suggests a shallow market for ORB products.

With the single exception of database tools, a higher percentage of programmers than executives expressed a need for all categories of OO products.

Exhibit V-1



Source: INPUT

B

Vendor Sales: Products

Eighteen percent of the vendors surveyed provided an integrated product that encompasses several or all of the individual product categories queried. Of the responding vendors, approximately one-third provided each of the specific OO products.

Such conformity suggests that presently there is no singular "hot" OO product that vendors are clamoring to market. This also demonstrates a lack of "one-stop shopping" opportunities for companies seeking to invest in OO products.

Only 50% of the vendors surveyed were able to provide information regarding a best-selling product. Of those responding, again no clear best

seller was apparent; 18% reported databases and class libraries, 13% reported CASE tools, visual design products, and languages, and 8% reported ORBs and objects. No vendors selected middleware as a top seller.

C
Need for OO Services

1. Executives

Of company executives surveyed, 70% expressed a need for one or more of the above OO services. Of those, 63% demonstrated a desire for OO services in three or more categories.

The need for training in OO technology by far commanded the greatest response from executives. Nearly 50% more executives expressed a need for this category than for the second most commonly selected service, consulting. Only a small number of executives specified Internet-related services.

2. Programmers

Seventy-three percent of the programmers surveyed expressed a need for one or more of the above OO vendor services. Of those, 67% demonstrated a desire for OO services in three or more categories.

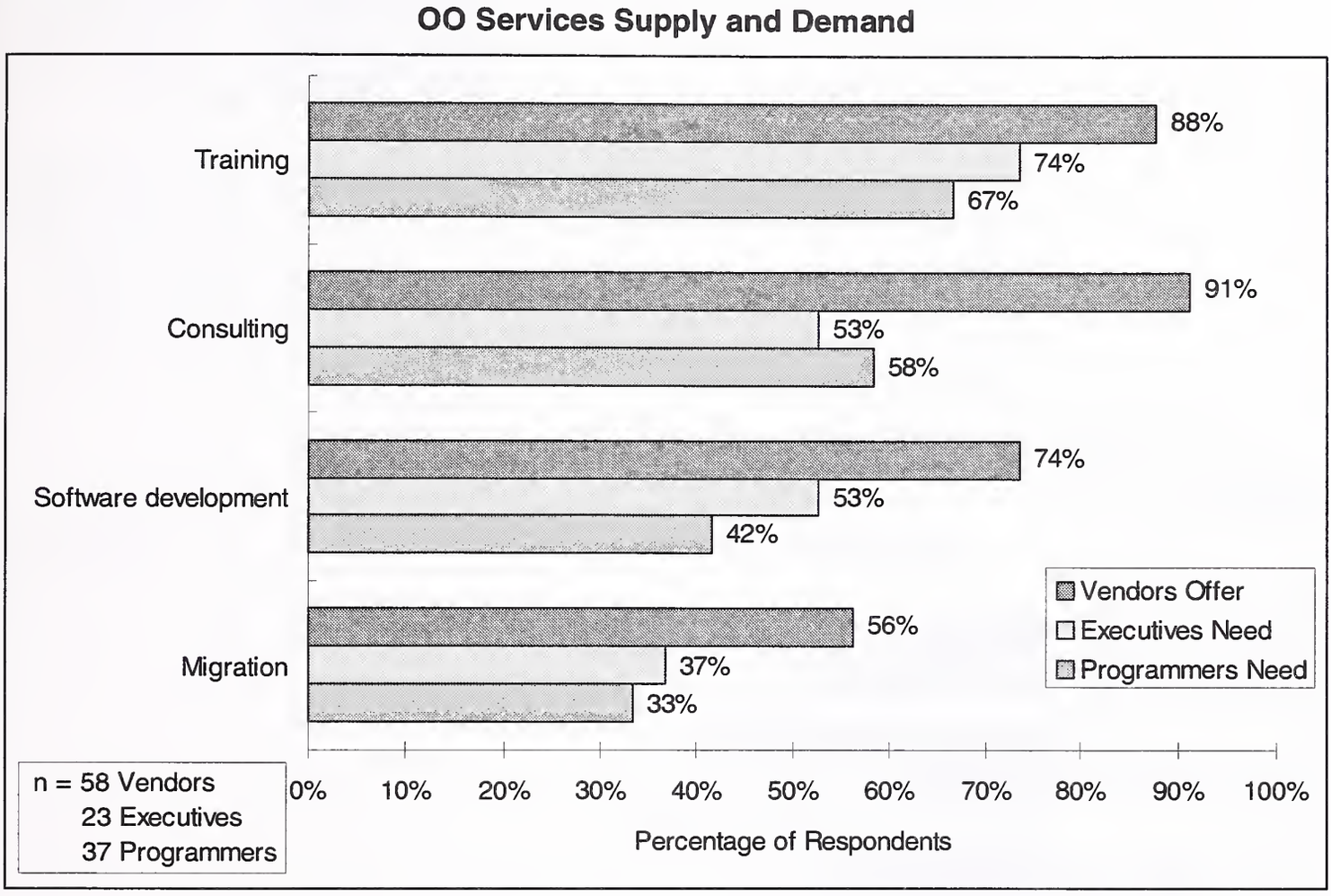
Of the six primary categories, training was also ranked highest by programmers. Following closely, the need for consulting services was identified by a larger percentage of programmers than executives. No programmers expressed an interest in Internet-related services, but showed a small interest in project management software.

D
Vendor Sales: Services

Of the vendors surveyed, 5% selected more than one "best seller" category and 9% could not determine or did not distinguish a best-selling service. The categories of consulting (39%), software development (32%), and training (21%) were favored significantly over migration and systems integration (4% apiece).

This bias is perhaps owing to the close interrelationships between the three services: Consulting often encompasses software development and/or training.

Exhibit V-2



Source: INPUT

E

1998 IS Budgets

INPUT estimates that 20%, or nearly $6 billion of the $29 billion 1996 U.S. professional services market, will be spent on object technology services. This figure is expected to grow to 75%-80% over the next five years.

Exhibit V-3 depicts a comparison of 1996 IS department budgets and those forecasted for 1998. The most dramatic shift will occur in the $1 million to $3 million range, shrinking by over a third from 44% in 1996 to 28% in 1998.

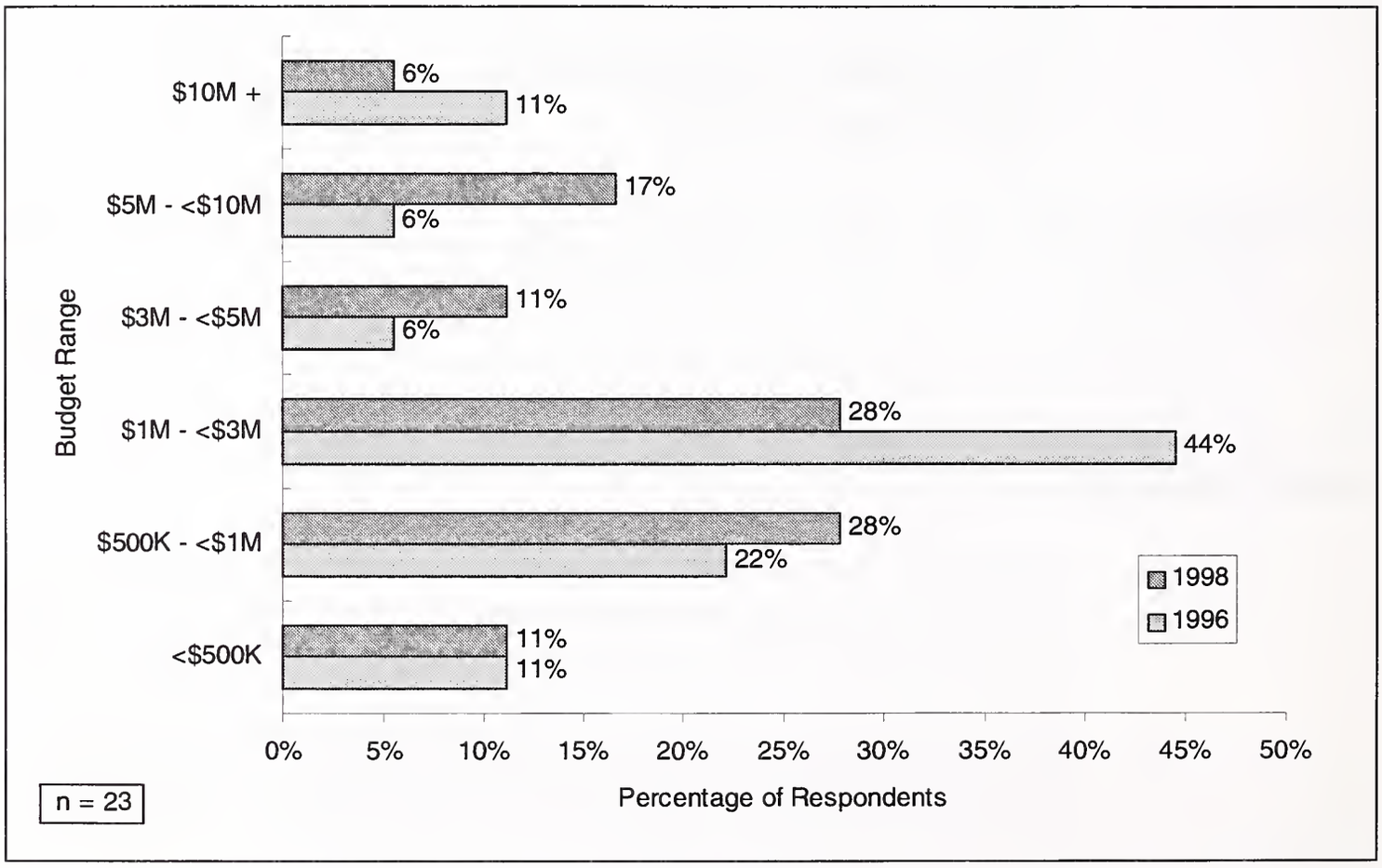
This budget range will remain a dominant category, but will be matched by expenditures in the $500,000 to $1 million range, which will grow marginally from 22% in 1996. The brunt of the shift has migrated upward, with the notable exception of IS departments that are expecting to spend in excess of $10 million being cut nearly in half.

This aside, budgets in the $3 million to $5 million range are expected to nearly double, from 6% to 11%, and those in the $5 million to $10 million range may triple, from 6% to 17%. The number of departments with budgets of less than $500,000 are expected to remain relatively unchanged.

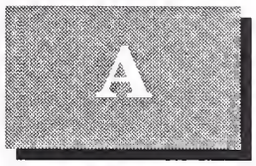
Approximately one-quarter of the departments surveyed anticipate budgets of $5 million or more in 1998 and three-quarters expect budgets of $5 million or less. Over half of the respondents predict a 1998 IS department budget range between $500,000 and $3 million. These points suggest that object-oriented expenditures will constitute an increasingly significant portion of these IS budgets.

Exhibit V-3

Executives' 1998 Forecasted IS Budgets



Source: INPUT



Definitions

This appendix provides definitions of common client/server terms and expressions. More definitions can be found in INPUT's *Definition of Terms*.

Glossary

Agent	An agent is a set of instructions that can carry out tasks automatically. It is usually written in a high-level language script and may run across a network to send messages or find information.
API	Application programming interface—This provides specifications for programmers.
Applet	A small application that may be embedded in another application; for example, Microsoft's MS Graph is a graph program used in the MS Excel spreadsheet and the MS PowerPoint presentation package.
Client	When used in C/S it refers to the computer platform accessed by a user, such as a PC, workstation, or PDA.
Component	Component refers to a software component, a piece of software with documented interfaces that a programmer can use to build an application.
CORBA	Common Object Request Broker Architecture.
Departmental server	A server priced as the minicomputers in INPUT's *Definition of Terms*. Examples are a high-end PC or an SMP UNIX server.
Development	Set of software used by programmers for developing

Environment	applications that typically consists of compilers, debuggers, visual editors, profilers, and performance optimizers.
Development Tools	Short for "application development tools."
Distributed System	A system that runs across multiple computers.
DLL	Dynamic Linked Library-A software component of precompiled code that can be linked into an application.
Enterprise Server	A server priced as a mainframe in INPUT's *Definition of Terms*. Examples are an IBM-compatible mainframe or a large SMP server of comparable price. These machines are often clustered.
Firewalls	Hardware and/or software solutions that prevent data from entering or leaving a network. They are most commonly used to protect corporate LANs from the Internet. They protect what may leave a corporate network and also what may enter it.
Framework	A specification or implementation of software that can be used to build an application. It may consist of classes and methods. Motif and the Common Object Request Broker Architecture (CORBA) are examples of frameworks.
Gateway	Software that connects one environment to another. It often translates formats and routes code from one application to another.
GUI	Graphical user interface—A windowing system like Microsoft Windows or X-Windows with Motif that displays graphical objects.
HTML	Hypertext markup language—A language for document formats, a dialect of SGML.
Http	High transport transmission protocol—A protocol to move information between a Web server and a client on the internet.

Internet	A publicly available network based on TCP/IP protocols that supports electronic mail, Web sites and other communications solutions.
Intranet	A private network that uses Internet technology; for example, an internal TCP/IP network with browsers and Web servers.
LAN	Local-area network.
MacOS	The operating system for the Apple Macintosh.
Messaging	A general term that describes communication that stores and forwards information. It may also support queues of objects waiting for an event in a network. An example of messaging software is electronic mail or software that supports on-line information services.
Microkernel	An OS architecture in which the system is built around a core set of software known as the microkernel. The microkernel need not necessarily be small. The Mach kernel used by NeXT is an example of a microkernel-based OS, as is Apple's Copland.
Middleware	In this report, middleware is connectivity software that links clients and servers. It is systems software. Further information can be found in INPUT's report, *Middleware: Is DCE the Answer?* Some companies include databases and visual development tools as middleware; these are not included in INPUT's definition of middleware.
Object Request Broker (ORB)	OMG terminology for the message-based communications interface between objects. An ORB provides the mechanism by which objects transparently make requests of, and receive responses from, other objects; the term has become commonly accepted, but not all products that perform these functions are called ORBs and not all ORBs meet OMG specifications.
On-line services	Services that provide users with information, like America Online, CompuServe, Minitel, NiftyServe, Dialog, and Lexis-Nexis.

OODBMS	Object-oriented database management system.
ORDBMS	Object-relational database management system.
Open systems	In this report, it describes systems that can run on multiple UNIX and/or Windows operating systems, rather than proprietary environments like VMS (even Open VMS) or MVS (even with POSIX compatibility).
OpenDoc	A component software standard managed by Component Integration Labs that was originally set up by Apple, IBM, and Novell.
Operating environment	Modern term for an operating system plus its application development tools.
OS	Operating system.
PDA	Personal digital assistant—A handheld small computer with personal address lists, organizer information, etc. An example is Apple's Newton.
Platform	This is the software or hardware that an application program runs on.
Port	Verb, as in *to port* or *porting*. Move software from one platform to another; for example if Windows NT is moved to run on Digital's Alpha-based computers, it is ported to run on the Alpha environment.
POSIX	A standard for operating systems to ensure some level of portability of software code that runs on it. Standards are published by X/Open.
Program	The term is meant to include a wide range of possible constructs, including scripts, loadable modules, etc., in addition to the traditional definition of an application or utility.
Reseller	An individual or company that resells a product. A reseller may or may not change the product to add value. See VAR.
ROM	Read-only memory—Used to store information that needs to be readily accessible in a computer. In a PDA it may contain the entire OS.

SGML	Standard generalized markup language—A language defined by IBM and others for document formats, it is used by the government and manufacturing organizations as part of the CALS standards.
Suites	Sets of applications or packages. Office suites typically consist of a word processor, a spreadsheet, and a database or electronic mail package.
Three-tier	A C/S architecture consisting of three logical parts of the system: a client, an application logic server, and a data/information server. This may be on three computers, or the application logic and data may reside on the same machine, but be logically separate. This model enables data sources and databases to be swapped in and out of the system more easily than in the two-tier model, where the data and application logic may both be stored in the same database.
Two-tier	A C/S architecture in which there is a client and a server. Business logic either resides in the client portion of the application software or in the server.
URL	Universal Resource Locator—A string that describes an entity on the Internet. For example, "http://www.input.com" describes the URL for INPUT's Web site.
VAR	Value-added reseller—A reseller that adds value to software or hardware by customizing it, adapting it for a specific market segment, integrating it, porting it to a new environment, or adding software to it.
Videoconferencing	Communications between two sites in which each site sees and hears information from the other. The information may be captured from a computer screen or a camera. It may use videoconferencing equipment or a camera on a PC. Communications may be via high-speed phone lines or private networks.
Visual Development	This is the software needed to build an application. It may

Tool	include a visual editor, a forms designer, a report writer, a compiler, an interpreter, a debugger, or a source code control system that enables programmers to share coding tasks.
WAN	Wide-area network.
Web	The World Wide Web.
Windows	Used in this report to refer to Microsoft's Windows if it starts with a capital letter. If it starts with a small letter then it may refer to any software that controls the windows on a computer screen. A window may also be the window seen on a computer screen.
Workgroup	A networked group of computers or people that share information. Typical sizes range from a few individuals to about 100 people.
WWW	The World Wide Web.

B

Organizations

Names and Addresses of Organizations

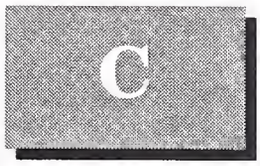
This section lists leading client/server consortia and standards organizations.

Exhibit B-1

Names and Addresses of Organizations

Company	Notes
Component Integration Laboratories, Inc. P.O. Box 61747 Sunnyvale, CA 94008 Tel: 408-864-0300 Fax: 408-864-0380	CI Labs is a nonprofit organization founded by IBM, Apple, Novell, WordPerfect, SunSoft, Taligent, and the XSoft division of Xerox to develop and promote the OpenDoc architecture.
The Object Management Group 492 Old Connecticut Path Framingham, MA 01701 Tel: 508-820-4300 http://www.omg.org	The OMG was founded in 1989 by a core group that included Hewlett-Packard, Data General, American Airlines, and 10 other companies specifically interested in promoting object-technology standards. It has a close working relationship with X/Open. It has also managed to attract major user organizations, telecommunications vendors, and systems integrators.
Object Database Management Group 13504 Clinton Place Burnsville, MN 55337 Tel: 612-953-7250 Fax: 612-397-7203 http://www.odmg.org	The ODMG was formed to create one standard interface that all object-oriented database vendors could support. The objective is application portability, as opposed to interoperability.
Open Software Foundation (OSF) 11 Cambridge Center Cambridge, MA 02142 Tel: 617-621-7300 Fax: 617-621-8700 http://www.osf.org	Recently announced Internet products for the World Wide Web. Currently undergoing a management transition.

(Blank)



Questionnaire

INPUT, a research and consulting company based in Mountain View, California, is conducting a study of business readiness for Y2000. I would like 10 minutes of your time so that we might understand your company's preparations in this area.

Your name and your company's name will no be released and all your answers will be kept confidential. We will send a complementary copy of the executive overview of this study as a thank you for your assistance.

Are you responsible for your company's IT preparation for the year 2000?

Y/N ___

If **YES** — Go to Question 1.

If **NO** — Who is the person responsible for this activity? _____

What is their position and telephone number?

Position _____ Telephone Number _____

Thank you for your assistance.

End of interview

Status of preparation

1. In preparation for Y2000, which of the following activities have you completed?

Activity	Completed
Working on obtaining upper mgmt. buy-in	
Someone made responsible for Y2000 readiness	
Developed plan of action	
Audited all applications	
Modified or changed applications	
Migration	
Testing	
Implemented revised applications	
Other	

1a. Comments on current status

Preferred Approach to Resolving the Issue

2. Please rate, on a scale of 1 to 5 (where 1 = will not use and 5 = will definitely use), how likely is it that you will use the following approaches to changing your applications?

Approach	Rating
Implement upgrade to existing package	
Modify existing custom software	
Rewrite existing application	
Build new custom application	
Implement new application package	
Outsourcing your IT department	
Outsource management of application code	
Contract for a disaster recovery service	
Other (1)	
Other (2)	

2a. Comments on approach to Y2000 issue

Critical Skill Requirements

3. Please rate, on a scale of 1 to 5 (where 1 = not important and 5 = very important), how important the availability of the following skills is to the success of your Y2000 preparations:

Skill	Rating
Project Management Expertise	
Y2000 Audit experience	
Implementation of package software	
Previous experience with Y2000 changes	
COBOL program development	
C language development	
Other language development (which one?)	
Other Skills (describe)	

3a. Comments on skill requirements

4. Who will provide the following functions for your Y2000 project? In-house (IH) or external service provider (ESP)?

Function	Provided by:
Project management	
Transition Methodology - the plan	
Inventory	
Assessment	
Planning	
Migration (rehosting, rewriting, replacing, etc.)	
Testing	
Implementation	
Other Functions (describe)	

4a. Comments on functions provision

5. Which of the following skills would be needed from an external service provider?

Skill	Use an ESP
Project Management	
Strategy Consulting	
Application Design and Development	
Test Planning and Design	
Network Planning and Design	
Data Migration/Database Design	
Testing	
Implementation (roll-out)	
Other Skill (describe)	

5a. Comments on external skills demand

Source of Funding for Y2000 preparation

6. How do you intend to fund the activities associated with Y2000 preparation?

Source	Y/N
Separately budgeted item	
Included in budgets of previously planned projects	
Reduce/cancel expenditure in other IT developments	
Increase/overrun previous budgets	
Reduce non/IT expenditure	
Other (describe)	
Do not know how it will be funded	

6a. Comments on source of Y2000 funding

Y2000 Tools

7. Have you used, or do you intend to use, any special tools to help you with your Y2000 preparations?

   Y/N___

7a. If "Yes", then which tools have you used and, on a scale of 1 to 5 (5 = very useful) how useful was effective was each tool?

Tool (If name of tool not known then identify its purpose)	**Rating of usefulness**

7b. Comments on Y2000 tools

Vendors

8. How would you rate the suitability or preference of the following types of vendors to provide assistance with your year 2000 plans? (scale 1 - 5, 5 = most suitable or preferred)

Vendor type	Rating
Y2000 Consultants: companies that evolved or were created to address the Y2000 issue	
Y2000 Tool Vendors: companies focusing primarily on providing tools to assist others in the Y2000 inventory, assessment, migration and testing	
Outsource/Off-shore Providers: companies that focus on migrating systems with large labor pools or semi-automated "factories"	
Systems Vendors: companies that offer both hardware/software solutions and professional services	
Professional Services Vendors: Y2000 extensions to existing services and partnerships with tool vendors	
Other Type of Vendors	

9. Have you used, or do you intend to use, any outside service vendors to help you with your Y2000 preparations?

   Y/N___

9a. If "Yes", then what did they do and who were the vendors?

Role	Vendor Name

9b. Comments on Y2000 service vendors

Estimate of cost

10. What is your current estimate of the cost, to your company, of fixing the Y2000 cost?

___ Do not know

___ Less than $100,000 ___ $1 million - $5 million

___ $100k - $500K ___ $5 million - $10 million

___ $500K - $1 million ___ Over $10 million

10a. Which of the following cost elements did you include in your estimate?

Cost elements included in estimate	Y/N	Percent of Total Cost
Internal staff		
Software package upgrade		
Cost of new application package		
External consultants and developers		
New hardware		
Education and training		
Other (describe)		

10b. Comments on the cost of preparing for Y2000

Timescale for completion

11. When do you think you will have all IT changes complete and implemented?

Already completed	_____	1998	_____
1996	_____	1999	_____
1997	_____	2000	_____

11a. What factors are most likely to disrupt your plans and cause you to miss your targets?

12. Do you have any other comments regarding the Y2000 issue and its resolution? For example, what lessons have you learned and what would be your advice to other companies when they are considering the Y2000 issue?

Lessons learned

Advice to others

Other comments

Thank you for your time and patience. We will send you a summary of the study as soon as it is available.

End of interview