Carnegie Mellon
Software Engineering Institute

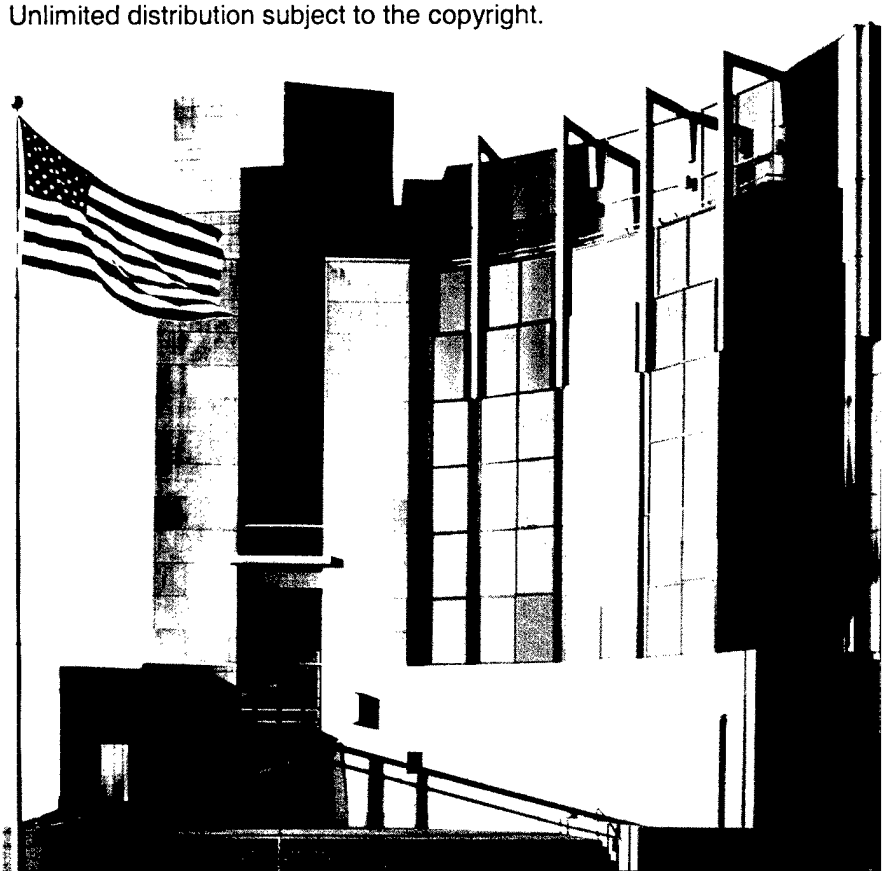
Volume I: Market Assessment of Component-Based Software Engineering

Len Bass
Charles Buhman
Santiago Comella-Dorda
Fred Long
John Robert
Robert Seacord
Kurt Wallnau

May 2000

Internal Research and Development

TECHNICAL NOTE
CMU/SEI-2001-TN-007



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Volume I: Market Assessment of Component-Based Software Engineering

Len Bass
Charles Buhman
Santiago Comella-Dorda
Fred Long
John Robert
Robert Seacord
Kurt Wallnau

May 2000

Internal Research and Development

TECHNICAL NOTE
CMU/SEI-2001-TN-007

printed 9/26/01 8:02 AM v2.0 / bw4le

Contents

List of Figures

Abstract

To assess the market for component-based software engineering, the Software Engineering Institute (SEI) studied industry trends in the use of software components. The study, conducted from September 1999 to February 2000, examined software components from both technical and business perspectives. The results of this study are summarized in the following technical notes and reports:

Volume I: Market Assessment of Component-Based Software Engineering

Volume II: Technical Concepts of Component-Based Software Engineering

Volume III: SEI Role in Component-Based Software Engineering

This technical note, Volume I, examines software component technology from a business perspective. It synthesizes the views of economists, industry analysts, information technology (IT) managers, and engineers. It presents evidence that software component technology is indeed being adopted by commercial industry. It also explains what lies behind the adoption of software component technology, and what industry expects from software component technology.

1 Introduction

Pick up any software or Information Technology (IT) magazine today and you will see the terms "software component," "component-based development," "componentware," etc. Is this just the latest technology fad or is there something more fundamental happening? Is there substance beyond the hype, and, if so, is there a genuine opportunity to advance the state of software engineering practice? If such an opportunity exists, can the Software Engineering Institute (SEI) play a role in accelerating these advances?

To address these questions, the SEI studied industry trends in the use of software components. The study, conducted from September 1999 to February 2000, examined software components from both a technical and business perspective. The results of this study are summarized in three reports:

Volume I: Market Assessment of Component-Based Software Engineering

Volume II: Technical Concepts of Component-Based Software Engineering

Volume III: SEI Role in Component-Based Software Engineering

This report, Volume I, examines software component technology from a business perspective. It synthesizes the views of economists, industry analysts, information technology managers, and engineers. It presents evidence that software component technology is indeed being adopted by commercial industry. It seeks to explain what lies behind the adoption of software component technology, and what industry expects from software component technology.

Volume II[1] examines software component technology from a computer science perspective. It synthesizes the views of leading researchers in software component technology, tempered by trends in commercial software component technology. It seeks to identify and relate the key technical concepts that underlie software component technology, and to provide a basis for principled discussion of the gaps between hype and practice and the potential role that the SEI might play to reduce these gaps.

Volume III will outline a suggested course of action for the SEI. It will describe key obstacles that inhibit the emergence of an engineering discipline for using software components, and the technical program needed to overcome these obstacles. It will also show that the technical program is feasible (although not without risk) and will explain why the SEI is the right organization to carry out the proposed work.

[1] http://www.sei.cmu.edu/publications/documents/00.reports/00tr008.html

A Note on Terminology

We would do well to define some basic terms before proceeding. For example, what are software components, and what do we mean by component-based development, software component technology, component-based software engineering, and so forth? In truth, we mean different things by these terms in Volume I than we do in Volumes II and III.

The objective of Volume II is, in effect, to give precise definitions (to the best practical extent possible) to the key concepts of component-based software, while Volume III uses these more precise definitions to describe a technical agenda for SEI work in this area. But for Volume I we are forced to employ a much broader category for software component technology. Why? Because industry does not speak consistently to the question of what a software component is. Some equate commercial-off-the-shelf (COTS) software packages with components. Some consider the use of some underlying technology such as Microsoft's COM to be the defining criterion for component. Quite apart from these conceptual categories is the question of size. Some consider components to be the small-scale equivalent of objects in object-oriented programs, while others consider components to be the large-grained equivalent of subsystems or larger.

There are many reasons for this conceptual dissonance, not the least of which is the need for technology vendors (and software researchers?) to differentiate their products, as well as the very natural generality of the term *component* itself. Regardless of the cause of this dissonance, we need to define our categories in a way that is sufficiently general to capture broad trends in component technology. For the purposes of Volume I, the following definitions are used:

- A software *component* is an implementation, in software, of some functionality. It is reused as-is in different applications, and accessed via an application-programming interface. It may, but need not be, sold as a commercial product. A software component is generally implemented by and for a particular component technology.
- Software *component technology* includes the software that provides a runtime environment for software components (sometimes called a component *framework*) as well as any other tools useful in designing, building, combining, or deploying components or applications built from components.
- *Component-based development* (CBD) involves the technical steps for designing and implementing software components, assembling systems from pre-built software components, and deploying assembled systems into their target environments.
- *Component-based software engineering* (CBSE) involves the practices needed to perform CBD in a repeatable way to build systems that have predictable properties.

Despite the generality of the above definitions, they are, when combined with our knowledge of software components, adequate to the task of a market assessment.

Outline of this Report

Section 2 describes in more detail the objectives of this report and the research method employed. Section 3 presents the key evidence of a shift toward software components. Section 4 describes the current and short-term IT environment and how this IT environment will create specific software needs within commercial and government organizations. Section 5 summarizes the benefits of software components and describes how component-based software development addresses specific commercial and government software needs. Section 6 characterizes the software component market and identifies key players. Section 7 describes the inhibitors to a software component market and highlights areas of possible SEI research.

2 Objective and Approach

2.1 Objective

The objective of the Market Assessment is to examine component-based software technology from a business and market perspective. The study's objectives are to describe or identify the

- potential size of component market
- dominant players and "emerging" organizations
- factors driving the adoption of software component technology
- perceived benefits of component-based development
- perceived inhibitors to the realization of component-based technology

2.2 Approach

Data was collected from a variety of sources including

- *literature survey*. A wide variety of articles (approximately 80) were reviewed.
- *previous research and market forecasts*. Third-party research organizations including Gartner, Ovum, and PriceWaterhouseCoopers were consulted.
- *Internet search*. We reviewed over a hundred sites of industry-leading component technology providers and consumers.
- *market surveys*. We conducted two market surveys (questionnaires):
  1. a Web survey with approximately 80 responses
  2. an email survey with approximately 50 responses
- *telephone interviews*. We spoke with approximately two dozen managers of various components-related organizations.
- *face-to-face interviews with key executives*. We conducted in-depth meetings with researchers and executives from the Theory Center, FLASHLINE.com, and Sterling software.

3 Indication of Shift to Components

Most leading industry analysts predict an increasingly prominent role for software component technology. For example, one Gartner Group analysis states

> *The onset of server-side* components *is inevitable; nearly every self-respecting application server vendor is planning to incorporate a* component model *in its product. The added abstraction and control that component models will bring will be a certain productivity benefit.* [Gartner 98b].

In report after report, predictions about the growth of software components technology are dramatic (if sometimes breathless) as we see in another statement from the Gartner Group:

> *By the year 2002, 70 percent of all new applications will be deployed using* component-based *application building blocks (0.8 probability).* [Gartner 00]

Several industry trends are motivating the adoption of software component technology. One notable trend is increasing reliance on various forms of Electronic Commerce (EC). That EC applications are at the forefront of this component trend is illustrated by another Gartner report:

> *By 2001, at least 60 percent of EC applications will be built using components (0.7 probability).* [Gartner 98a]

These are strong statements, but the Gartner Group is not alone in its enthusiasm for a growing component market. Figure 1 summarizes software component market size estimates from four major market researchers, Gartner, Giga, Ovum, and PriceWaterhouseCoopers. Although these estimates vary considerably in absolute terms, they share the same general trend lines. The key point to see is that four major market researchers agree that a shift toward component-based software engineering is not only predicted, ***it has already started***.

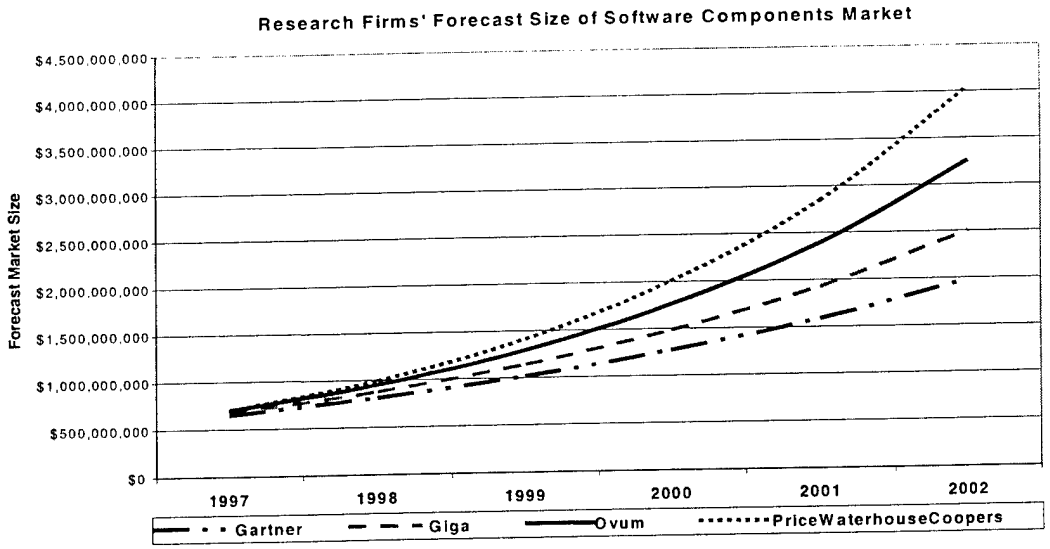


Figure 1: Projections of Overall Growth in Component Market

Evidence that the component market is already here and on the rise can be found in the growing customer volume of two companies that broker software components. Flashline.com and ComponentSource are component brokerage companies enjoying substantial success selling ActiveX, JavaBean, and Enterprise JavaBean (EJB) components. According to executives interviewed for this report, Flashline.com, for example, has four thousand sessions per day on its Web site and, at the time of publication, the company is growing at 20% per month.

More evidence of an existing and growing component market is found in the extraordinary success of companies that build software components. The Theory Center builds EJB components for electronic commerce applications and sells a suite of components (through companies such as Flashline.com) to customers who want to quickly assemble an EC application. This small startup company numbering 12 people when they visited the SEI in March 1999 was recently purchased by BEA for one-hundred million dollars after growing to 42 people in November 1999 [Theory 99]. There is certainly a business case for building software components!

As organizations build applications from components, they develop techniques and processes based upon their experience. These techniques for designing and building applications using software components have evolved into a new engineering discipline called *component-based development* (CBD) or *component-based software engineering* (CBSE). There is even a market for selling CBD/CBSE methods and experience. For example, MTW Corporation created a course called "Progressive Method" that describes how to construct applications using Sterling components. The Progressive Method is not for sale per se, but is rather the

foundation for MTW's consultancy business. which is entirely restricted to the development of component-based systems.

The significance of software components and the need for software component research have reached a national level of visibility and importance. According to the President's Information Technology Advisory Committee (PITAC) *Report to the President*, the construction and availability of libraries of *certifiably* robust, specified, modeled, and tested software components would greatly aid the development of new software. The PITAC report recommends funding fundamental research in software development methods and component technologies to provide the scientific and technological foundations needed for a software component industry [PITAC 99].

The excitement about software components can also be seen in technically oriented literature, where, arguably, "hype" is balanced by thoughtful discourse. The July 1999 issue of IEEE Computer magazine has software components as the cover feature and consists of 6 articles with 40-plus pages. Moreover, this was not the only issue devoted to the topic of software components. The lead article, entitled "Component-Based Development: From Buzz to Spark" describes a spark as occurring when the entire software industry "suddenly and synchronously 'clicks' on an idea and determines that it's going to change forever the way we do our business" [Meyer 99]. Meyer goes on to say

> *It's still early to tell for sure, but component-based development will most likely become a spark—if we get it right.*

This quote hints at critical issues inhibiting CBD. The article describes the primary inhibitor as quality or specifying and certifying component properties. We will discuss this and other inhibitors later in this report. For now we merely note that serious obstacles lie between the nearly delirious expectations of some industry analysts and the actual state of the technology. Nonetheless, the very breadth of interest in software component technology is the strongest argument in favor of taking it very seriously.

4 Information Technology Environment

The growing information technology market provides the context for the booming demand for software component technology. As IT becomes more critical to business performance, demand for more and better quality software becomes more pronounced, and software frequently becomes the limiting factor in corporate competitiveness. Software component technology is widely seen as the best (if not only) means of achieving the gains in programmer productivity, system flexibility, and overall system quality required by the IT revolution.

4.1 The IT Revolution in Global Economics

IT is an essential decision tool for competitive advantage in the global market. An eminent authority on the subject, Federal Reserve Chairman Alan Greenspan, testified before the Joint Economic Committee (U.S. Congress on June 14, 1999) that current U.S. and global economic growth is rooted in advances in IT. Chairman Greenspan specifically credited IT:

> *An economy that twenty years ago seemed to have seen its better days, is displaying a remarkable run of economic growth that appears to have its roots in ongoing advances in technology. [I]nnovations in information technology—so-called IT—have begun to alter the manner in which we do business and create value, often in ways that were not readily foreseeable even five years ago* [Greenspan 99].

Chairman Greenspan's statement also indicates that new ways of using IT are being discovered, and that these discoveries are forcing economists to reevaluate old theories of the limiting factors of sustained economic growth. Moreover this trend is not limited to the commercial world. IT has also become essential in government and defense applications. In a 1999 presentation entitled "How IT is Changing Our Defense Strategy," VADM Arthur Cebrowski, President of the Naval War College, described an emerging theory of war that uses network-centric warfare and information superiority [Cebrowski 99].

The result of this IT revolution—the critical role of IT to competitiveness combined with the relentless search for new and different ways of using IT to create competitive advantage—is creating extraordinary demand for more, different, and better IT systems. In particular, the IT revolution implies the following demand:

- Radically reduced time-to-market for IT resources, as the early IT innovator reaps the greatest reward.
- Significant gains in programmer productivity, as the national capacity to produce IT resources rapidly outstrips the supply of software developers.
- Systems that are adaptable, so that changes in business processes and policies can be quickly realized in underlying IT resources.
- Systems that are reliable, secure, and scalable, as might be expected from IT resources that are mission critical and yet must operate in a distributed and often exposed environment.

All of these "demands" existed before the IT revolution, but the IT revolution is creating vast market incentives to satisfy these demands—and, as will be seen later, market data indicates that component technology is, for now, the beneficiary of many of these incentives.

4.2 E-Commerce

E-commerce is dramatically changing the way people and organizations purchase commercial products and services. Cars are sold electronically; bank customers access their account information from their homes; manufacturers buy their supplies through Web auction sites. Personal e-commerce organizations such as Amazon.com are complimented by new business-to-business e-commerce organizations such as FreeMarkets.com.

E-commerce is also becoming a central part of government procurement, including Department of Defense (DoD) acquisition. All major DoD services and the Defense Logistics Agency have moved at least part of their acquisition process to e-commerce. A list of sites can be found at http://dodbusopps.com.

Peter F. Drucker argues that e-commerce on the Internet is changing the way products are distributed, in a way no less revolutionary than the railroad's dramatic impact on distribution in an industrial society [Drucker 99]:

> *It is something that practically no one foresaw or, indeed, even talked about ten or fifteen years ago: e-commerce—that is, the explosive emergence of the Internet as a major, perhaps eventually the major, worldwide distribution channel for goods, for services, and surprisingly, for managerial and professional jobs.*

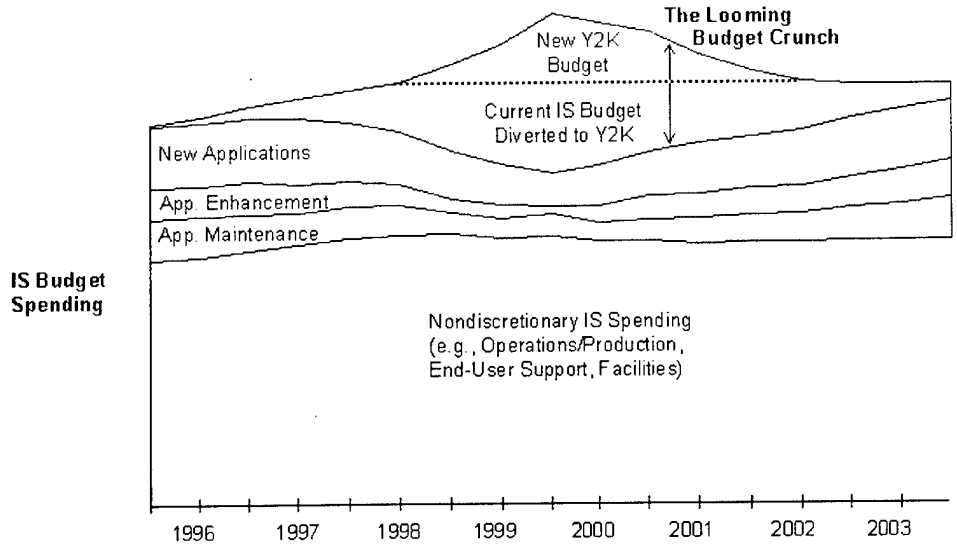
The Internet's explosive growth and rapidly changing Internet technologies are increasing the motivation for software productivity. Specifically, e-commerce application consumers and developers want

- *scalability* to manage rapid growth
- *rapid development* to address aggressive global competition
- *cheaper development* to manage cost
- *disposability* to manage rapidly changing technologies

In the competitive e-commerce market, reducing costs and improving system quality attributes are critical. For e-commerce, systems *must* be architected with a specific focus on quality attributes such as reliability, performance, and security. Commonalties of services like security and transactions must be exploited, time to market reduced, flexibility and cycle time optimized, and legacy data connected with this new distribution channel.

4.3 The Y2K Backlog

IT-dependent organizations worldwide are now awakening to a Y2K hangover of massive proportions. For the past several years, IT spending has been diverted, in large measure, to Y2K risk reduction. As a consequence, the growing demand for more and different kinds of software, discussed above, has been constrained by available IT resources—funding and developer talent. But as Y2K recedes in prominence, Chief Technology Officers (CTOs) will increasingly divert their attention—and funding resources—to the growing IT backlog. The figure below depicts this as a budget "crunch," where post Y2K spending drops in absolute terms while funding for new application development and enhancement grows [Gartner 99d].



Y2K - Spend Anyway	8 %	11 %	11 %	9 %	9 %	5 %	4 %
Y2K - Diverted Spending	0	7	14	11	6	2	0
Sub-Total	**8**	**18**	**25**	**20**	**15**	**7**	**4**
Y2K - New Funding Req.	0	2	11	7	4	1	0
Total	**8 %**	**20 %**	**36 %**	**27 %**	**19 %**	**8 %**	**4 %**

Figure 2: Y2K Budget

The proper interpretation of this data, however, is not to assume the emergence of a budget crunch, but rather a budget boom, as IT organizations re-apply previously diverted Y2K resources to satisfy application development backlog. Indeed, one major vendor of component technology disclosed to us that their company forecasted a dramatic drop in component technology spending by their customers leading up to Y2K, to be followed by a significant upturn in investment beginning in the second quarter of 2000.

Although it is too early to conclude that this forecast is bearing fruit, we can say that Y2K spending has placed an artificial damper on IT modernization, and that this damper will soon be more folklore than market factor. This view is shared by Gartner in yet another strategic planning advisory [Gartner 99c]. It refers to the "Post Year-2000 Tsunami," which presents the claim that

> *...major ESPs (External Service Providers) are not just working harder in the technical space: they are working smarter. They are increasingly using generic application templates, and are focused on technical architectures based on reusable components.*

It appears, then, that adoption of software components will, if anything, accelerate as discretionary IT funds become available.

4.4 Summary

The central role of IT in economic competitiveness has been recognized at the highest levels of U.S. economic policy making, in corporate boardrooms, and in the offices of CTOs. The demand for more IT systems, and IT innovation, is creating unprecedented demand for new software. This demand, when coupled with fast-breaking technologies such as the Web and new modes of business based on e-commerce, is expected to increase significantly. Overall, industry is looking for approaches to building IT resources that

- Improve programmer productivity and "reduce time to market" for IT resources.
- Produce systems that are flexible enough to withstand technology and business changes.
- Produce systems that deliver excellent performance and are scalable, secure, and robust.

Much of the interest in software component technology is linked to the perception that software components can meet these demands. In fact, since most of the concepts that underlie software component technology are not new (as is discussed in Volume II), the growth of the software component technology marketplace can be seen as an economic response rather than as an innovative breakthrough.

The next section details market perceptions regarding the adoption of component technology and provides evidence (such as it exists) of its utility.

5 Why Components?

Component-based software engineering specifically addresses productivity issues that are facing commercial and government organizations in the emerging IT environment. In this section, we describe the perceived benefits of component-based software and present evidence collected during the market assessment.

5.1 Components Improve Programmer Productivity

The argument that software components will improve programmer productivity is an old one with roots in the study of software reuse. The argument, put simply, is that productivity and quality increase if programmers can *reuse* software. There are many techniques to achieve reuse, including reuse of design patterns, use of application-specific programming languages coupled with program generators, and so forth. Reuse of software *components* also has obvious appeal, especially insofar as components (as we define them in this report) are opaque implementations that must be reused "as is."[1]

Evidence that this reuse argument applies to software component technology is scarce, but does exist. For example, a Gartner study noted that companies that adopt component-based development see a 50% improvement in programming cost [Williams 99]. However, not everyone is optimistic that such productivity gains will occur immediately. One market prediction suggests that only 30% of first-generation corporate component investments will generate expected productivity improvements, due to undefined goals and conflicting component contexts that inhibit reuse [Gartner 99b]. Part of the challenge is for organizations to define a suitable scope of reuse for recouping initial investments in component technologies—a situation not unfamiliar in past reuse "war" stories.

Consistent with our earlier assertion that component technology is an economic rather than technological response, we find a market response to the question of defining optimal scopes for component reuse. Interestingly, this market response also follows the contours of previous research in software reuse—namely, the market is producing *domain-specific* components and families of components. In the commercial world, domain-specific is most often interpreted as a vertical market niche. For example, the Financial Industry Group at EDS, a global leader in information solutions for the financial services industry, offers a set of EDS Engineered Business Components for corporate systems developers in the financial services sector. The Theory Center likewise has found success in focusing their component "product line" to the development of e-commerce resources. (EDS and the Theory Center are

[1] There are ample studies that demonstrate the diminishing economic incentive for reuse where the reused software undergoes even minor modification.

discussed in more detail, later.) Domain-specific component technologies are also emerging in the healthcare, insurance, and geographic data industries.

A reasonable conclusion is that the increased size and maturity of the software market are providing sufficient economic incentive for investment in domain-specific software reuse—in particular, the domains have become sufficiently large and coherent to foster reuse industries. We can also conclude that a component-based approach to software reuse has won out over alternative technology approaches to software reuse, such as Fourth General Languages (4GLs) and object-oriented frameworks. Some possible explanations for this "victory" are discussed below.

5.2 Components Reduce Time to Market

Closely related to improved productivity is reduced time to market. Although productivity improvements do not always reduce time to market, there is evidence that they do where software component technology is used. While productivity can be attributed to reuse of previously developed software (what Will Tracz describes as "rewarding theft over honest toil"), software component technology leads to reduced time to market for at least two additional reasons:

> First, component product lines for vertical niches such as e-commerce applications treat system building as a matter of configuring and interconnecting pre-built components within the context of a well-defined application architecture. (By application architecture, we mean well-defined types of components and assemblies of components.) For example, the Theory Center provides a family of e-commerce components called JumpStart based on EJB. The Theory Center asserts that JumpStart allows assembly of routine e-commerce applications in as few as 30 days. The recent success of the Theory Center in the marketplace is eloquent if oblique evidence of the validity of this assertion.
>
> Second, software component technology raises the level of abstraction of system building by packaging various sources of complexity into a component framework, and by allowing a separation of skills in the component-based development process. Component frameworks often manage many complex and low-level aspects of inter-component communication and resource management (e.g., persistence, transactions, error reporting, security), allowing developers to concentrate on application functionality. Separation of skills is partially illustrated by the framework/component distinction between infrastructure builder and application builder, although other separations can also be discussed.

Sun's EJB, for example, illustrates separation of skills by differentiating (and providing distinct services for) component developers, application assemblers, application deployers, and framework administrators. The SEI-developed WaterBeans component technology illustrates the interplay of higher-abstraction and separation of skills [Plakosh 99].

WaterBeans is a visual component framework developed by the SEI for the Environmental Protection Agency (EPA) to support water quality modeling and simulation. EPA domain experts, without special programming skills, can quickly and visually compose water quality simulations from WaterBean components. All aspects of inter-component communication and coordination are "built into" the component framework.

5.3 Components and System Quality Attributes

There is evidence that component technology, when coupled with component product lines, improves programmer productivity and reduces overall time to market. There is little evidence, however, that component technology satisfies the other demands of the IT revolution, namely demand for systems that are flexible, scalable, secure, and so forth—the so-called system *quality attributes*. Indeed, for the near term it seems that technology consumers have made an apparent Faustian bargain: improved productivity and time to market in exchange for a vague and uneasy sense of "trust" that commercial components and component frameworks will deliver the desired quality attributes.

Consumers are not unaware of the potential risks inherent in this bargain, as will become apparent in our later discussion of inhibitors to component-based software engineering. We will see that the number one concern of consumers *after* the availability of a stable market in component technology is the availability of *certified* software components and the means to obtain *confidence* that these components will *lead to* the desired *system quality attributes*.

5.4 Components Provide a Basis for Reuse Commerce

Perhaps the biggest reason for commercial industry to adopt component technology as the mechanism to achieve software reuse is that components are a convenient way to package value. This value comes in two forms: components provide a flexible boundary for economy of scope, and components can be easily distributed (in the sense of shipping goods from one point to another, rather than in the sense of "distributed system").

By scope we mean, loosely, the functional boundary of a component—what it implements, as opposed to all the things it doesn't implement. Economy of scope is the *value* to consumers or producers of components derived from a particular scope. Components can capture a narrow scope (they can be "fine grained") or a wide scope (they can be "coarse grained"), and the scope can be easily altered to suit changing market conditions. In contrast, 4GLs, object-oriented frameworks, and other reuse approaches are much more rigid regarding their economy of scope. From the consumer's point of view, too, components are convenient in that they are more "bite sized" than 4GLs and so forth. Components are designed to be units of substitution; therefore, there is, in theory, a lower switchover cost between components than, for example, between 4GLs. In addition, the economy of scope of components has direct utility—they can be applied directly to build a system, in contrast to design patterns or other more abstract forms of packaged reuse, which require significant transformation before providing utility.

Coupled with a flexible means for defining a component economy of scope is the ease with which components can be distributed. The clear boundaries and concrete nature of components make their sale and distribution simpler than is the case for the fuzzier and more abstract reuse packages, such as design patterns. This is particularly apropos given the emergence of the Web as a distribution channel. The Web provides component marketplaces where consumers can comparison shop different components from different vendors. These marketplaces are operated by new businesses called component brokers who wish to connect component providers with component consumers. Companies that broker components, such as Gamelan, Flashline.com, and ComponentSource, use the Web for all customer contact and product distribution.

5.5 Summary

To date, the success of component technology in satisfying the demand for software generated by the IT revolution hinges largely on the ability of components to support domain-specific software reuse. Improvements in programmer productivity and reduction in overall system time-to-market are all attributable primarily to the ability of software component technology to capture reusable application logic; secondarily, to its ability to abstract away many low-level details of system assembly. Most important, though, is that component technology provides a foundation for commerce in reusable assets called components and component frameworks. In the next section we describe the various market sectors that have emerged to support commerce in software components and component frameworks.

Before proceeding, however, it is important to recall that software component technology has not yet demonstrated an ability to predictably meet the requirements for scale, robustness, and performance that IT-bound organizations are facing. These deficiencies will need to be addressed if component-based technology is to succeed in the long term and to become a software engineering discipline.

6 Component Economy

In this section, we characterize the software component technology market and provide some forecasts of market growth.

6.1 Component Market

Component developers are finding business opportunity in building components for a variety of applications. This opportunity for selling components has created component markets that began with small-grained graphical user interface (GUI) controls and is moving to medium and large-grained server-side components.

Published research is in agreement that the marketplace for software components is undergoing dramatic growth. Just how much, however, is a subject for debate, as illustrated in Figure 3. It shows market predictions from four major market analysis organizations. Much of the disagreement on market size is a function of how the various research firms define "component." For example, some research firms have included "mega-components" which are created by encapsulating existing large applications (such as BAAN or SAP) in a component "wrapper." Other research firms do not include these types of components. The result is a significant discrepancy on the size of the marketplace even though all predict substantial growth—in the range of 25% to 40% per year.

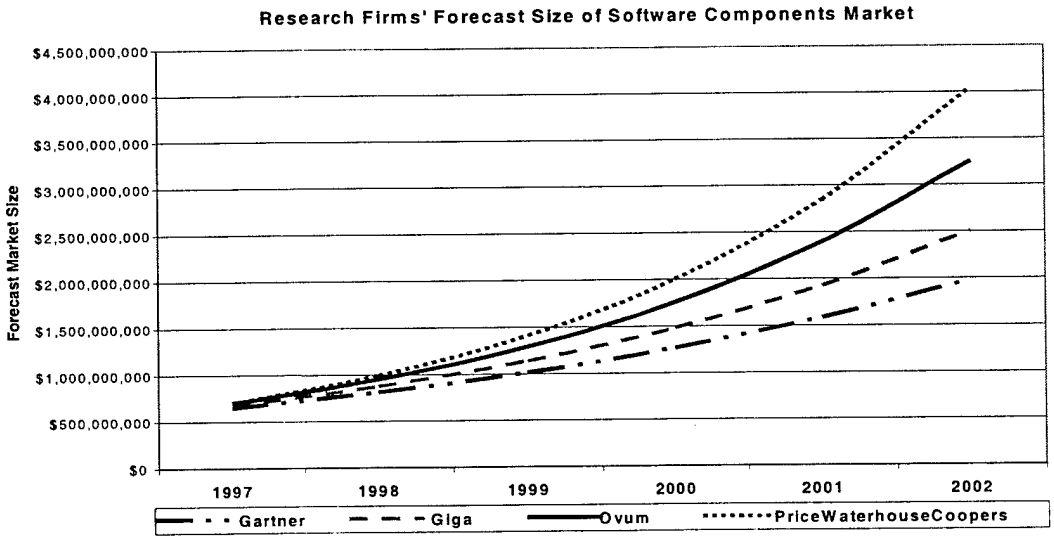


Figure 3: Projections of Overall Growth in Component Market

A question related to the overall growth of the component market concerns the distribution of this growth across different component technologies. A study by the International Data Corporation (IDC), shown in Figure 4, predicts that Microsoft will continue to dominate competitors. (The Microsoft component market is projected to be twice that of all rival component technologies *combined.*) But when considered in absolute rather than relative terms, the same study shows that the market for alternative ("niche") component technologies (i.e., not Sun or Microsoft) will be quite large—on the order of $100 million by 2001.

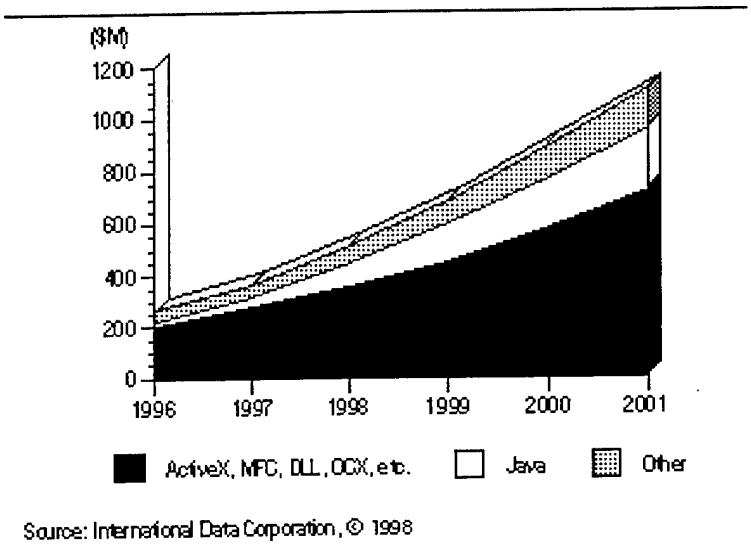


Figure 4: Software Components Revenue by Architecture

Nonetheless, the picture that emerges from these projections is a large and growing component market that is restricted to two major component standards, one based in Java and the other based in Microsoft COM and its extensions.

6.2 Product Lines of Components Market

With increasing frequency, components are not sold as stand-alone units, but rather as a family of related and interacting components. These component families constitute *product lines* for vertical markets as insurance, financial services, or communication.

An example of a component product line is EDS Engineered Business Components. The Financial Industry Group at EDS, a global leader in information solutions for the financial services industry, entered the software components market by offering a suite of financial components to corporate systems developers. By purchasing software components from EDS, these developers can significantly reduce their development costs and resources, and speed time-to-market while still meeting their organization's specific business requirements. Another example of a component product line is the Theory Center, which provides a suite of

EJB e-commerce components called JumpStart. JumpStart includes 80 EJB eBusiness Smart Components that are designed to address 50-75% of an organization's e-business requirements out of the box. The Theory Center also models all components using UML and can generate some code from UML to overcome interoperability problems between EJB platforms.

6.3 Infrastructure Market

The infrastructure market is the set of standards, component platforms, and component tools that support component-based development. Component infrastructure lays the foundations for developing component-based systems and is rapidly changing as CBD concepts evolve.

6.3.1 Infrastructure Standards

Standards play an important role by consolidating the infrastructure market and providing markets for component developers. The extent and stability of these standards is an indication of market maturity [Gartner 99a]. The primary sources of infrastructure standards are the Microsoft Component Object Model (COM) standards, Sun Microsystems (Java, EJB, JavaBeans standards), and the Object Management Group (Common Object Request Broker Architecture [CORBA] standards).

6.3.1.1 Microsoft

Microsoft's COM has become one of the most popular component standards in the industry. Microsoft evolved this standard into Distributed Component Object Model (DCOM) to support distributed components and distributed applications. This basic model continues to evolve with COM+, which, among other things, supports a new naming and load balancing service.

The Microsoft component models are both flexible and restricted. They are flexible in that they support a variety of programming languages (Java, C++, VisualBasic) but they are restricted in that they only run on Windows platforms. Nevertheless, these component technologies support a large and growing component market.

6.3.1.2 Sun Microsystems

Sun Microsystems and aligned vendors have developed two component specifications: JavaBeans and Enterprise JavaBeans (EJB). Both JavaBeans and EJB are promoted as extensions to the Java programming language and therefore provide a platform-neutral programming solution that is intimately tied to the Java standards.

JavaBeans is a component specification largely intended for creating GUI controls in the Java programming language. JavaBeans are a logical extension to Motif, OpenLook, and other GUI components that Sun has been involved with in the past. Perhaps the most novel concept implemented in JavaBeans is introspection. External applications can, at runtime,

determine the application-programming interface for a JavaBean. This results in a very flexible component model and one, moreover, that reduces coupling between components and therefore facilitates the growth of independent third-party commodity components (i.e., components not necessarily embedded in a product line of components).

Enterprise JavaBeans is a server-side infrastructure standard that was released in 1998 and has been evolving with support from selected Sun partners. EJB defines a runtime infrastructure that includes support for transactions, security, and persistence. In addition to the specification, Sun released a reference implementation of EJB including the source code. Over 30 vendors are providing commercial implementations of the EJB server that implements the EJB standard. (A complete list can be found at http://java.sun.com/products/ejb/tools1.html.)

6.3.1.3 OMG

The Object Management Group (OMG) made the CORBA 2 specification available in 1996; this defines a middleware infrastructure that can be used to integrate components. A number of products have been produced that implement the CORBA 2 specification, including Inprise VisiBroker and IONA Orbix. CORBA 2 also defines the Internet Inter-ORB Protocol (IIOP) that is being widely used as an underlying protocol in EJB and remote method invocation (RMI) to support inter-operability between different application servers.

The CORBA 3 specification has recently extended the previous CORBA 2 specification with a component architecture, called CORBAcomponents. The three major parts of CORBAcomponents are

- a container environment that packages transactionality, security, and persistence, and provides interface and event resolution
- integration with Enterprise JavaBeans
- a software distribution format that enables development of the CORBAcomponent software marketplace

A key goal of CORBA 3 is to leverage the skills of business programmers so they can now produce business applications that acquire these necessary attributes automatically. There are currently no implementations of CORBA 3, so the market success of this component standard has yet to be determined.

6.3.2 Component Platforms and Application Servers

A platform is an implementation of an infrastructure standard that enables the execution of components. These platforms, when targeted to enterprise IT computing, are often called *application servers*. Component technology providers such as Sun, Microsoft, and IBM spend tremendous resources to stake their claims in this growing component market.

Many component platform vendors exist in the EJB and CORBA markets. Ovum predicts in "Application Servers: Creating the Web-Enabled Enterprise" that by 2004, the market for application servers (a subset of the infrastructure market) will be worth nearly $17 billion [Ovum 99]. Some application server vendors are betting that interoperability in this segmented market is important, and are providing adapters and translators to support multiple component models.

Microsoft is different from other component platform providers in that it developed the component standard *and* produced the component platform. There are only a handful of component platforms (Windows 95, 98, NT, 2000) for COM, DCOM, or ActiveX components, but the pervasive use of Windows has created a tremendous market for components based on these standards.

6.3.3 Component Tools Market

Similar to the object-oriented (OO) technology market, the component market is witnessing a growing selection of tools to support component-based development and component management. A 1998 International Data Corporation (IDC) white paper [Garone 98] presented the data in Figure 5 as evidence of the expected rapid growth of the component-based development market.

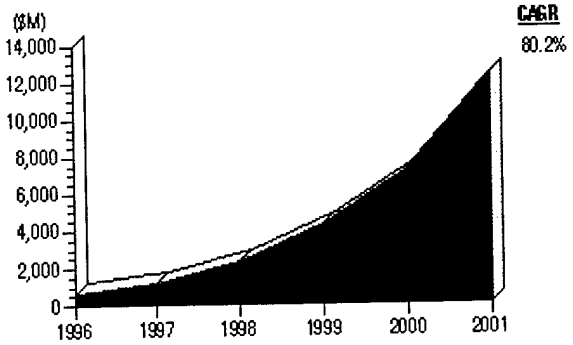


Figure 5: Component-Based Development Revenue, 1996-2001

The same IDC white paper also presented the data in Figure 6 as evidence of the expected growth of the component management market. These studies were written in 1998 and predicted that by 2000, the component-based development and component management markets would double and quadruple, respectively. Although we cannot confirm whether these predictions have come to pass, we can state that Sterling Software, a major vendor of software component technology, forecasts analogous growth in the tools market for component-based development.

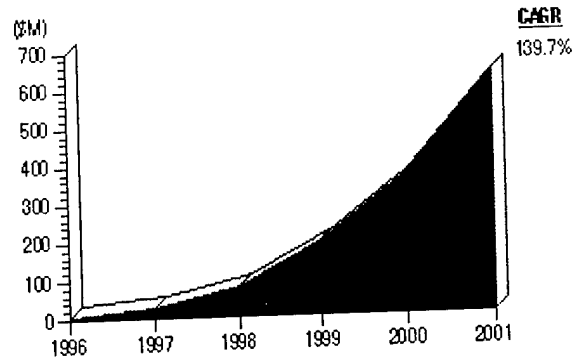


Figure 6: Component Management Revenue

However, we also know from discussions with Sterling that the tools market is fickle. Development tools are sometimes bundled with or developed by infrastructure providers so they can distinguish their infrastructure products in a somewhat standardized market. However, both Sterling and Select Tools (from Select Software Tools) are examples of vendors that are maintaining some independence from the infrastructure market.

6.4 Consulting/Integrator Market

In component-based development (CBD), system integrators select available components and develop custom components, and assemble these components into applications. These component-based development consultancies base their competitive advantage on well-defined CBD processes and deep familiarity with a particular component technology and toolset. The previously mentioned MTW Corporation typifies this market. Systems integrators may also build systems from product lines. These consultancies base their competitive advantage on the strength of their product lines. This is the business model adopted by the Theory Center.

Component consulting is growing as application consumers demand more component-based systems. "In 1999, IT organizations will spend an estimated $53.3 billion for worldwide systems integration services," according to Ed Acly, Director of Middleware Research with IDC. By 2002, IDC projects this spending will increase to nearly $76 billion a year. Vendors such as Sterling, Pricewaterhouse, and EDS lead this market niche.

6.5 Brokerage Market

Brokers are companies focused solely on providing a marketplace for software components. For example, they may provide application developers with mechanisms to help find and acquire software components. Two firms in particular, Flashline.com and ComponentSource,

have established a significant presence and appear to be viable, at least in the short term. Both of these companies provide an Internet-based 'trading-post' for software components developed by third parties.

Flashline.com distributes general components for the World Wide Web (WWW), information management, and user interfaces. Specialized components have also been offered recently. For example, Enterprise JavaBeans from the Theory Center provides a complete suite of functionality for e-commerce applications. In addition to these specialized components, Flashline.com is now distributing tools and component middleware for component-based development, including EJB application servers, graphical editors, and application debuggers.

ComponentSource provides a range of components, from general search and spelling functions to more specific business functions such as credit card authorization and barcode components. Components are categorized by vendor and function while a search engine can search all available component descriptions. Many of these components have freely available evaluation versions that can be downloaded for immediate testing.

It is too soon to tell whether component brokerage will succeed, but the growth reported by Flashline.com (20% per month) is promising.

6.6 Third-Party Component Certification Market

Software certification is not new. The National Security Agency (NSA) has long certified software for specific security attributes. What is new is that the growing commerce in software (not necessarily "component-based" as we have used that term in this report) has created a business opportunity for organizations to *certify* that software conforms to certain standards, provides particular qualities of service, or exhibits certain quality attributes. It is worth noting that this interest in certification is not restricted to government agencies or to security qualities. For example, Underwriters Laboratories (UL) recently announced a new standard for certifying the safety of software that is embedded in programmable components [UL 99]. The certification tests for failures due to software design faults, coding faults, and timing faults.

Of the markets discussed above, third-party certification is the most speculative. While our surveys indicate that the *lack* of certified (or "trusted") components is a significant concern for consumers of component technology (see next section), business and technical obstacles also are inhibiting the emergence of certification standards. On the business side, there is still considerable flux in infrastructure standards, and component adoption is still in its infancy. Most adopting organizations are concerned with getting something working first, and consider system and component quality attributes to be secondary and tertiary considerations. More severe still are the technical challenges posed by questions such as these: what qualities need to be certified, how are these qualities described and measured, and how are system qualities predicated on these qualities? Unless these and related questions can be answered, it

is unlikely that a robust market in third-party component certification will emerge, no matter how much consumers of software component technology may desire it.

6.7 Summary

An entire industry is growing up to support software component technology. As in any industry there are specialized niches that all contribute in a value chain from constituent parts to finished products. These niches include the components themselves, sold either as commodities or as product lines; infrastructure; tools; integration services; brokerage services; and certification services. Significant growth is projected or demonstrated in each of these niches except component certification, which remains speculative.

We now turn our attention to potential inhibitors to the successful adoption of software component technology.

7 Inhibitors

A distinct set of inhibitors to adopting software component technology emerged from SEI-conducted surveys and interviews of earlier adopters of software component technology. A summary from a Web survey of component adopters is presented in Figure 7.

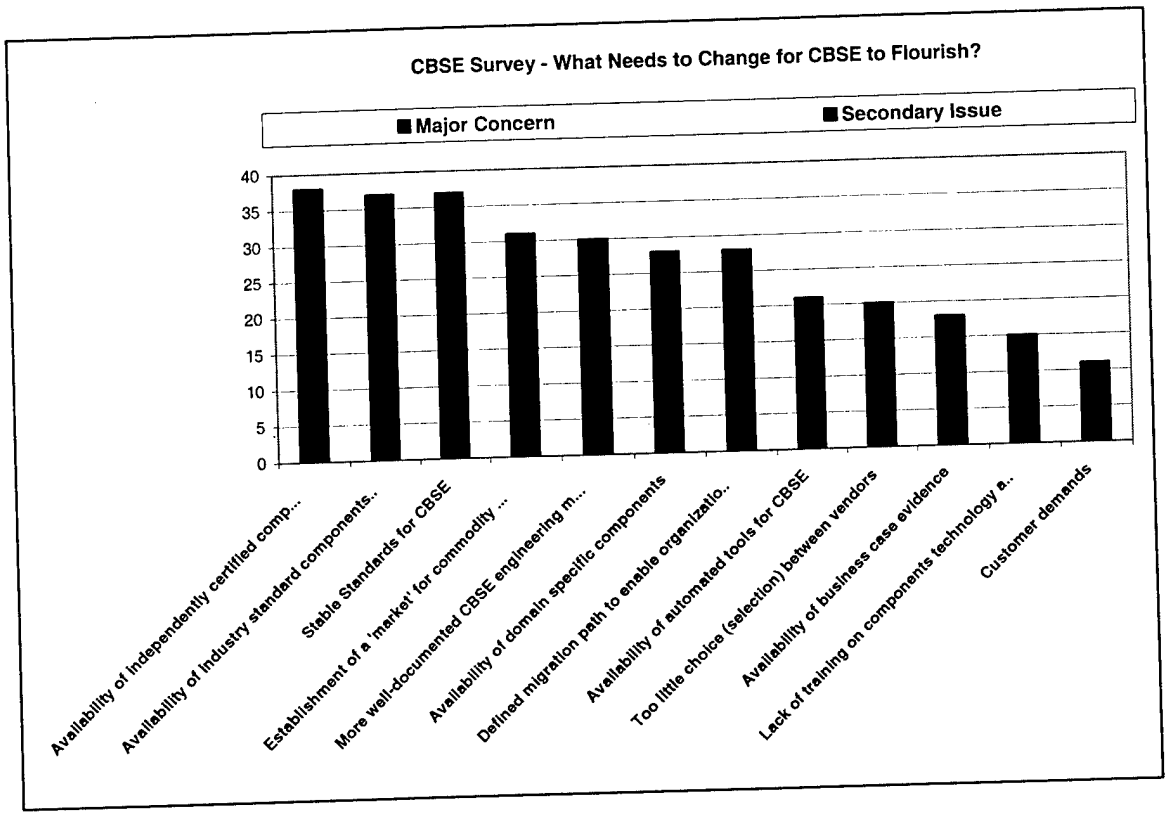


Figure 7: Summary of Survey Responses

From this data and from our interviews, we conclude that *the market perceives* the following key inhibitors, presented here in *decreasing* order of importance:

- lack of *available* components
- lack of *stable standards* for component technology
- lack of *certified* components
- lack of an *engineering method* to consistently produce quality systems from components

We discuss each in turn. Along with inhibitors we indicate where the SEI has a potential role in overcoming the inhibitor. These potential roles are mentioned only in passing here, but will be taken up in more detail in Volume III.

7.1 Lack of Available Components

Over 30% of the respondents claimed that a major concern was the lack of *industry standard* components. An additional 20% claimed that the lack of *domain-specific* components also inhibited the success of software component technology. Both responses reflect the same consumer orientation: For component-based development to succeed, components must be aligned to a vertical application domain.

This is an interesting perception when juxtaposed with the earlier cited projections for the growth of the component market. It suggests that either the component market is not yet sufficiently broad and/or deep, or that consumers are having difficulty in locating these components. Possible causes for the former might be the fragmentation and instability of the component infrastructure market, or simply that we are still too early in the projected growth cycle for the component market to have achieved critical mass. A possible cause for the latter might be the still-immature distribution channels for software components. It is likely that the component market, if growth projections are satisfied, will be too unwieldy for traditional advertising and distribution channels.

7.2 Lack of Stable Component Technology Standards

A perception shared by most survey and interview respondents was that software component technology is still immature. The adverse implications are expressed in terms of uneven quality of infrastructure products, the instability of these products (i.e., in the features they provide and how these features are implemented), and the standards that underlie infrastructure products. In the survey above, almost 30% of the respondents cited instability of software component standards as a major concern. As a result, consumers are reluctant to invest in component technology for fear of incurring undesirable "switchover" costs as new releases of infrastructure products and standards emerge.

Finding ways to address this inhibitor is a perplexing problem. As we know from our experience in the SEI COTS-Based Systems Initiative, those market forces that are generating software component technology—a technology that depends upon component standards—are also forcing vendors to differentiate their products and standards to achieve competitive advantage. Moreover, the process of differentiation is static. As new "differentiating" features become successful, they are copied by competing vendors. This forces a new round of product differentiation, and hence, instability.

7.3 Lack of Certified Components

One surprising result of our surveys and interviews is the perception that a lack of *independently certified* components is an inhibitor to the success of software component technology. This, too, is reflected in Figure 7, where certified components become the most prominent concern *after* stable component standards and a component market have been established (27% cite this as a secondary concern, while the next most significant concern is

stable standards). This concern is amplified when combined with the lack of component-based software engineering techniques that deliver predictable properties (see the next inhibitor).

The concern for certified components reflects a natural progression of concerns: *first* demonstrate that it is possible to build and sustain a component-based system at all, and *then* improve the overall quality of components and consumer trust in these components. Nevertheless, there are considerable technical (and business) inhibitors to achieving a market in independently certified software components.

7.4 Lack of Component-Based Engineering Methods

Most respondents to the SEI surveys and interviews tacitly accepted the view that component-based development requires its own set of engineering practices, although there was no general agreement about just what these practices are; this represents the nub of the problem. As noted earlier, some component technologies such as Sun's EJB introduce new roles in the development process. These roles and their contribution to the development activity are not accounted for in conventional software process maturity models. Many organizations want to adopt software component technology, but are afraid to do so precisely because there is an implied switchover cost from old to new engineering practices, and, moreover, it is not clear what these new practices are.

A close examination of the results of the SEI surveys and interviews focuses attention on a different aspect of this perceived inhibitor (i.e., lack of component-based methods): the particular quality attribute requirements placed on software component technology by the ever more demanding IT environment. This survey result is summarized in Figure 8.

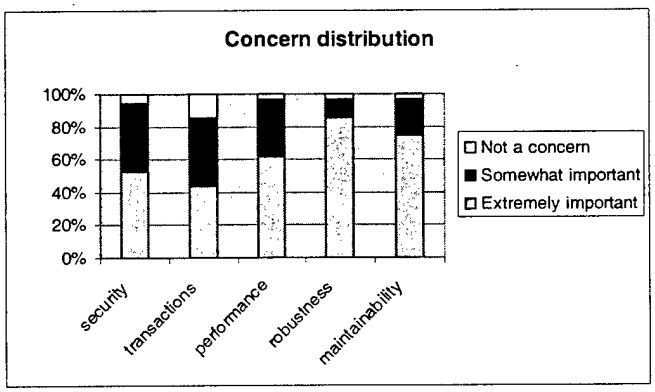


Figure 8: SEI Survey—Concern About System Quality Attribute

When asked about their more detailed concerns, very few respondents were indifferent. More than 80% claimed that system robustness (i.e., resilience to failure) was extremely important. Maintainability was a close second at over 75%, followed by performance, security, and transactional consistency.

These high levels of concern are mismatched with current component-based technology. The constituent products of a component-based development activity—components and frameworks—are complex with under-specified or unspecified quality attributes. Moreover, there are few methods and tools for reasoning about the properties of assemblies of components, assuming the properties of components are even known (which, currently, they are not).

Taken together—the fuzziness of the overall component-based development process and the lack of engineering technique for predicting the properties of component assemblies—these inhibitors constitute a significant risk to the adoption of software component technology, and a good opportunity for SEI leadership. This is especially true if these inhibitors can be addressed in tandem with those arising from the lack of certified components:

- The SEI might develop process guidelines and, ultimately, reduce these to key process areas for component-based software engineering. These process guidelines would be based on proven industry methods, such as the (now proprietary) MTW methods, or Rational's Unified method, or even D'Souza's Catalysis method.
- The SEI might develop engineering techniques for "predictable assembly," i.e., assembly of components having known properties and prediction of the assembled (sub)-system properties from component properties. This is in general a complex and currently unsolvable problem. However, certain aspects of software component technology, such as strong links between component frameworks and software architecture (discussed in depth in Volume II) make progress in this area feasible.

7.5 Summary

We have identified four principle inhibitors to successful industry adoption of software component technology. To be sure, we have not mentioned all of the usual inhibitors to the adoption of any new technology—cultural resistance (to reuse, for example), lack of personal incentives (conflicts with existing Capability Maturity Model® [CMM®]-based models, for example) and so forth. Instead, we have focused on the top four inhibitors, as revealed by a market-oriented study of software component technology:

1) lack of components

2) lack of stable standards

3) lack of certified components

4) lack of component-based software engineering methods

The SEI can play a role in overcoming all of these inhibitors. However, SEI prospects are best if applied to 3) and 4) together, as we will discuss at length in Volume III.

® Capability Maturity Model and CMM are registered in the U.S. Patent & Trademark Office.

8 Conclusions

There is substantial evidence that software component technology is emerging as a major factor in how systems are being, and will be, built for the foreseeable future. Leading market analysts predict growth in the software component industry, even if they do not agree on the rate of this growth.

The adoption of software component technology is being driven by fundamental changes in the role of IT in economic competitiveness. This new role for IT, noted by Greenspan, Drucker, and others, is placing unprecedented demands on the providers of IT resources. It is the IT providers who are turning to software component technology as the most promising way of meeting demands for increased programmer productivity, reduced time to market for IT resources, and improved system quality. Moreover, with Y2K largely resolved, IT organizations now have the funding and personnel resources ready to devote to the IT modernization backlog caused by the Y2K "crisis."

There is evidence of the emergence of a component-technology industry. Component technology market niches are becoming increasingly evident and well defined, including commodity components, product lines of components, infrastructure (frameworks and tools), component brokers, and component-based development consultants. Vendors are filling these niches, and, based on our surveys, are finding the markets competitive but promising.

Despite these positive signs, there are challenges to the sustained growth and success of software component technology. Consumers are impatient to reap the benefits promised by the technology, and are not seeking simply a new way of writing programs. First and foremost, they want more components that are useful in their lines of business. Second, they want the component infrastructure standards to stabilize so that disruptive changes in infrastructure products are minimized, and so that developers can write to a stable platform.

Assuming that these mechanistic aspects of component-based development can be resolved, there are additional (and, ultimately, more consequential) risks to the success of software component technology. These are found in the lack of proven component-based software engineering techniques. Adopting organizations are unsure about the impact component technology will have on existing development processes. Just as critical is the loss of predictability of system quality attributes that comes from relying exclusively on third-party components and component frameworks at a time when there is a growing premium placed on precisely these qualities.

The message, then, is that commercial industry is adopting software component technology and that, for now, all indicators point to continued growth. On the other hand, there are risks

that, if not addressed, will slow this adoption considerably. The SEI can play an effective role in mitigating some of these risks.

In Volume II, we describe in detail the technical concepts that underlie software component technology. It serves as a prelude for Volume III, which will propose a plan of action for the SEI to take in furthering a component-based software engineering discipline.

References

[Cebrowski 99] Cebrowski, Arthur. VADM "Keynote Address: How IT is Changing Our Defense Strategy." *Connecting Technology* (Fall 99).

[Drucker 99] Drucker, Peter F. "Beyond the Information Revolution." *The Atlantic Monthly* (Oct. 1999).

[Garone 98] Garone, Steve "Managing Component-Based Development: SELECT Software Tools." Framingham, MA:, International Data Corporation, 1998.

[Gartner 98a] Shih, C. & Satterthwaite, R. "MS's EC Component Platform; Plug and Play, but Not Today." *Gartner Group Strategic Analysis Report.* Stamford, CT: Gartner Group, June 1998.

[Gartner 98b] Percy, A. "The New Impedance Mismatch." *Gartner Group Research Note.* Stamford, CT: Gartner Group, November 1998.

[Gartner 99a] Natis, Y.; Pezzini, M. & Schulte, R. "Middleware Deployment Trends: Survey of Real-World Enterprise Applications." *Gartner Group Strategic Analysis Report.* Stamford, CT: Gartner Group, April 1999.

[Gartner 99b] Jones, N. "Why Your Component Investments May Never Pay Back." *Gartner Group Strategic Analysis Report.* Stamford, CT: Gartner Group, November 1999.

[Gartner 99c] Kyte, A. "The Post Year-2000 Tsunami Will Swamp In-House AD." *Gartner Group Strategic Planning Assumption Research Not.* Stamford, CT: Gartner Group, July 1999.

[Gartner 99d] Guptill, B.; Stewart, B.; Marcoccio, L.; Potter, K. & Claps, C. "1998 IT Spending and Staffing Survey Results." *Gartner Group Strategic Analysis Report.* Stamford, CT: Gartner Group, April 1,1999.

[Gartner 00] Phipps,D. "CBD: A Path to Uniform Component Design." *Gartner Group Strategic Analysis Report.* Stamford, CT: Gartner Group, Jan. 2000.

[Greenspan 99] Remarks by Chairman Alan Greenspan, "High-Tech Industry in the U.S. Economy." *Testimony before the Joint Economic Committee*, U.S. Congress, June 14, 1999.

[Kiely 98] Kiely, Don. "The Component Edge—An Industrywide Move To Component-Based Development Holds the Promise of Massive Productivity Gains." *Information Week* (April 1998). Available WWW. URL: <http://www.techweb.com/se/directlink.cgi?IWK19980413S0001>.

[Meyer 99] Meyer, B. & Mingins, C. "Component-Based Development: From Buzz to Spark." *IEEE Computer* (July 1999): 35-37.

[Ovum 99] Ovum update, "Application Server Market a Hit." Issue 7, October-December 1999.

[PITAC 99] President's Information Technology Advisory Committee, Presentation on Software Research. (1999). Available WWW. URL: <http://www.ccic.gov/ac/report>.

[Plakosh 99] Plakosh, Daniel; Smith, Dennis: & Wallnau, Kurt. *Builder's Guide for WaterBeans Components*, (CMU/SEI-99-TR-024, ADA373154) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available WWW. URL: <http://www.sei.cmu.edu/publications/documents/99.reports/99tr024/99tr024abstract.html>.

[Theory 99] Theory Center, press release dated November 10, 1999, "BEA Acquires the Theory Center and its Industry-Leading Portfolio of Java Components for Building E-Commerce Solutions."

[UL 99] Underwriters Laboratories Inc., *UL Launches Software Component Recognition Program.* Available WWW. URL: <http://www.ul.com/about/newsrel/comptest.htm> (1999).

[Williams 99] Williams, J. "Distributed Components and the Internet." *Application Development Trends* (December 1999): 79-80.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	May 2000	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Volume I: Market Assessment of Component-Based Software Engineering	F19628-00-C-0003

6. AUTHOR(S)

Len Bass, Charles Buhman, Santiago Comella-Dorda, Fred Long, John Robert, Robert Seacord, Kurt Wallnau

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2001-TN-007

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

To assess the market for component-based software engineering, the Software Engineering Institute (SEI) studied industry trends in the use of software components. The study, conducted from September 1999 to February 2000, examined software components from both technical and business perspectives. The results of this study are summarized in the following technical notes and reports:

Volume I: Market Assessment of Component-Based Software Engineering

Volume II: Technical Concepts of Component-Based Software Engineering

Volume III: SEI Role in Component-Based Software Engineering

This technical note, Volume I, examines software component technology from a business perspective. It synthesizes the views of economists, industry analysts, information technology (IT) managers, and engineers. It presents evidence that software component technology is indeed being adopted by commercial industry. It also explains what lies behind the adoption of software component technology, and what industry expects from software component technology.

14. SUBJECT TERMS	15. NUMBER OF PAGES
component-based software engineering, software components, component-based software engineering methods	41

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL