The Return on Investment on Commercial off-the-shelf (COTS) Software Components

Preliminary Study Results

Date: August 27, 2002

Author: Chris Brooke, ComponentSource

www.componentsource.com

Email: saveit@componentsource.com

US Headquarters

ComponentSource
3391 Town Point Drive NW
Suite 350
Kennesaw, GA 30144-7079
USA

Tel: (770) 250-6100
Fax: (770) 250-6199
International: +1 (770) 250-6100

European Headquarters

ComponentSource
30 Greyfriars Road,
Reading,
Berkshire RG1 1PE
United Kingdom

Tel: 0118 958 1111
Fax: 0118 958 9999
International: +44 118 958 1111

Contents

Introduction 1
- Why did we conduct this study? 1
- What benefits does this bring? 1
- How did we obtain and compile this data? 2

Key Assumptions 2
- **Measuring ROI on COTS components** 2
- Source Lines of Code 2
- Function Points 3
- Interface Points 3

- **Methods Used to Determine ROI** 4
- Development Time 4
- Development Cost 4
- Reuse Utilization 4

Analysis 5
- Table of sample data from study 5

Conclusion 6

Introduction

Corporate IT departments are under growing pressure to cut costs and speed development of their software applications while consistently improving quality. Increasingly, they are looking to reuse initiatives to help make this happen. The internal reuse of software assets brings many advantages to the table, including reduced redundancy by not "reinventing the wheel". However, the reuse of internally developed components does not go far enough to solve one of the most significant software development challenges: leveraging your developers' core competency – or, more accurately, requiring developers to venture outside of their core competency by learning how to code functionality that is already available via tried, tested and debugged third-party components.

For years now, the supply of commercial off-the-shelf software (COTS) components has grown significantly, with thousands of COTS components available today on the open market. These components allow developers to integrate complex functionality into their applications without requiring a steep learning curve. They are built on the same industry standards that developers use to create components internally, and as such, can be easily integrated into an organization's applications. In the past the return on investment (ROI) value of COTS components has been elusive to purchasers. This study for the first time shows how investing in an expert-built COTS component or Web service – which range in price from several hundred to several thousand dollars on the open market – may offset the development costs of applications to the tune of millions of dollars.

Why did we conduct this study?

While it seems obvious that the price paid for a COTS component ought to be less than the cost of equivalent in-house development, the actual cost of COTS components in terms of effort to develop and encapsulate has never been clear. Until now, the metrics for COTS Components has normally been hidden from purchasers. This provides tremendous value to the end user and industry and offers the measurable business rationale for "buy before you build", from a COTS component perspective. This is the first time that a study of this kind has ever been undertaken. Our relationship with component vendors internationally, places ComponentSource in a unique position to be able to conduct it.

We are publishing the raw data on our public marketplace for our half a million strong developer user base so that they are able to justify the buy versus build decision to management.

This data, moreover, is of importance to our SAVE-IT™[1] customers who need to be able to justify the acquisition and reuse of COTS components. The same metrics may be used to assign value to their internally-built components and they are able to forecast cost savings/cost avoidance from reusing software assets from both internal and external sources.

What benefits does this bring?

Component usage is fundamentally a supply-driven process. However, until a sufficient supply of internally sourced components becomes available within an organization, COTS component content can be critical to reuse adoption, hitting your project's early financial targets, and sustaining that effort over time.

The timeframe to stockpile the necessary critical mass of component content from internal efforts can stall reuse behavior, and prevent the required time-to-market and cost avoidance gains needed to achieve short term ROI. Historically, reuse initiatives fall short of breakeven for the first 18 to 24 months. Because of economic pressure, shifts in priorities, and changes in policy it is difficult to sustain the reuse efforts that occur beyond these early timeframes. By supplementing internal components with COTS components, reuse efforts can reach breakeven or positive ROI within a shortened financial horizon. This is because of the relative

cost/price advantage of bought versus built components.

In the long term, the marketplace at large will always possess an exponentially greater capacity to grow a categorically broad and functionally rich component supply more rapidly than any single organization could with its own resources. Structuring COTS component usage as an ongoing supplement to internal reuse efforts leverages this supply of marketplace intellectual property, thus expanding the organization's ability to deliver solutions without increasing headcount.

Accurate measurement of ROI is a critical success factor for any reuse initiative. It is, therefore, vital that organizations be able to measure the impact that COTS components will have on their overall project costs and time to market.

How did we obtain and compile this data?

As the market leader for reusable components, ComponentSource has partnerships with over 700 component vendors. We have leveraged these partnerships to collect the source lines of code (SLOC) by development environment behind each of their component products. Furthermore, a prerequisite information requirement for new component vendors that we bring to the market is that they give a line of code count behind their products. Using SLOC, we then apply industry metrics to estimate development time displaced in person months and cost avoidance as a dollar amount.

This study is two-thirds complete, with data still being collected.

With over 9,000 reusable COTS components and Web services currently available on the open market, this extensive study evolved as part of the work that ComponentSource was undertaking with corporate customers of its large-scale reuse solution SAVE-IT, many of which apply the same metrics used to assign value to COTS components to assign value to their internally-built components. This in turn enables them to forecast cost savings and cost avoidance from reusing software assets from both internal and external sources.

Key Assumptions

Measuring ROI on COTS Components (Metrics)

There are several methods used throughout the software development industry for measuring software productivity and reuse effectiveness. Of these, the three most commonly used by organizations are: source lines of code (SLOC), function points, and interface points. In this section, we look at these three methods and discuss how they are typically used.

a) Source Lines of Code (SLOC)

It is commonly accepted within the development community that for each development language, there is a predictable cost for each line of source code completed, debugged, and integrated into an application. When applied to COTS components, the data needed to determine value are development language and SLOC. The development language is readily known. As a matter of simple compatibility, vendors of COTS components include this information with their components when they are placed onto the open market.

Until now, vendors have not made information on SLOC available. As discussed in the introduction of this white paper, ComponentSource is leveraging its relationships with over 700 component

vendors - whose products comprise over 9,000 “best-of-breed” components on the open market- to fill in this gap, thus enabling effective measurement of reuse ROI by consumers of COTS components.

The following guidelines were used by the vendors for counting SLOC:

- Count functioning lines of code.
- Don’t count remarks/comments.
- Don’t count resource files/interface definition files.
- Don’t count conditional compilation lines that would not get compiled in the release executable.
- Don’t count standard header files provided by operating systems or the compiler.
- Do include header files if they include macros for lines of code.

These guidelines provide a solid level of consistency, allowing credible comparisons between multiple products from a variety of COTS component vendors.

b) Function Points

Function points were first utilized by IBM as a means to measure program volume[2]. Each function point represents a unit of functionality. This can be anything from calculating simple interest in a financial application, to performing complex parsing of data for formatting into a report. While this can be useful when measuring internal reuse - since the level of complexity for each function point is known - it is difficult to determine the work displacement for a COTS component, where the complexity of each individual function point is ***not*** known. When trying to determine the suitability of a component for use in an enterprise application, relying on function points is less than optimal.

c) Interface Points

Interface points represent specific interfaces that are exposed by the component and can be executed from the parent application. These include properties, methods and events. Each interface point may contain several function points, each containing possibly hundreds of lines of code. As with function points, interface points can be very useful when applied to internally developed components. However, since COTS components are generally designed for maximum flexibility, interface points are not able to provide the appropriate level of granularity for effective measurement.

Consider, for example, an enterprise business component that performs financial calculations. It could very well have only a few interface points, with each one encapsulating complex operations involving hundreds of lines of code. Conversely, it could expose individual methods to break this functionality into smaller, more specific operations. The size of the component remains the same, but its value in terms of work displaced would be considerably higher if packaged in this way.

Methods Used to Determine ROI

Of the methods outlined above, ComponentSource's study is based upon Source Lines of Code, as well as industry averages[3] for metrics such as person months to develop and deploy 1,000 lines of code by development environment, the cost per month of a salaried developer inclusive of office overheads and – given the fact that most commercial off-the-shelf components are feature-rich – the estimated percentage of a COTS component that may be used as part of an application. This study takes into account published work from the Software Engineering Institute (SEI) and Cost Xpert. It also includes input from expert organizations such as Software Productivity Research, Inc (SPR) which has specialized in Software Metrics, Software Estimation and Industry Benchmarking over the past 18 years. SPR has an accumulated knowledgebase of industry data and has studied over 10,000 projects of varying types.

The most important measurable statistic needed to perform an ROI analysis is the SLOC. From this we are able to use industry metrics to decipher the cost savings encapsulated in each component. The calculations were based on the following metrics:

SLOC * %Used = SLOC Displaced
SLOC Displaced * .0035 = Time Avoided (Person Months)
Time Avoided (Person Months) * Developer Cost Per Month = Cost Avoided

Cost Avoided - Single Developer License Price Of COTS = ROI

- SLOC - Source Lines of Code of the COTS Component.
- %Used - The percentage of the component that will actually be used (i.e. displace actual work).
- Time Avoided (Person Months) - Based upon the industry average of 3.5 person months per 1,000 lines of code.

This formula is explained in further detail in the following sub-sections.

a) Development Time

First we estimate the person-months required to develop 1,000 lines of code. This varies according to the development environment that you are in. The industry average is 3.5 person months to develop 1,000 lines of code in VC++/C++. Most of the data collected so far is for components developed in the VC++/C++ environment. This assumption may be changed according to the development environment.

b) Development Cost

The cost per month to employ one developer is around $10,000 – this is the overall cost to business to keep a salaried developer employed with office overheads. This assumption may be changed, for example a smaller organization may measure this cost at $5,000 per month.

c) Reuse Utilization

As these are feature-rich components, we assume 10 percent usage of a COTS component. For example, a grid or reporting component may have a variety of optional features that wouldn't necessarily be used. On the other hand, certain granular components – such as e-mail address validation or address de-duplication – are likely to have most or indeed all of their features utilized in an application. The metrics allow for this to be changed to any percentage value. Even if component usage is changed to 1 percent, savings are still very high.

Analysis

Utilizing the methods described in our assumptions, reuse metrics may be applied to COTS components in our public marketplace. The table below provides a subset of the components analyzed within the scope of the study, and demonstrates the consistency of the metric analysis across a range of products from different vendors. Of particular interest is the diversity of functionality represented in the preliminary results. As illustrated by the sample data below, a large percentage of these COTS components encapsulate specific business processes that are in particular demand in enterprise applications today.

Table 1 - Sample of study results

Component	SLOC	Language	% Component Used	SLOC Avoided	Time Avoided (Person Months)	Cost Avoided	I Developer License	ROI (x:1)	Component Type
Sax ActiveX Zip Objects	31,000	VC++	10%	3,100	11	$108,500	$399	272	Data Compression Components
Xceed Zip Compression Library	72,773	VC++	10%	7,277	25	$254,706	$300	849	Data Compression Components
Dart PowerTCP Zip Compression Tool	29,994	VC++	10%	2,999	10	$104,979	$249	422	Data Compression Components
Desaware StorageTools	131,000	C++,	10%	13,100	46	$458,500	$199	2,304	Data Storage Components
Dart PowerTCP Mail Tool	44,991	VC++	10%	4,499	16	$157,469	$499	316	Email Components
Xceed Encryption Library	42,338	VC++	10%	4,234	15	$148,183	$300	494	Encryption Components
Dart PowerTCP SSL Tool	123,843	VC++	10%	12,384	43	$433,451	$999	434	Encryption Components
Desaware File Property Component	11,000	C++	10%	1,100	4	$38,500	$79	487	File Handling Components
Data Dynamics #Grid	750,000	VC++	10%	75,000	263	$2,625,000	$249	10,542	Grid Components
LEADTOOLS Document Imaging	2,466,899	C, C++	10%	246,690	863	$8,634,147	$1,995	4,328	Imaging Components
Xceed Absolute Packager	13,817	Delphi	10%	1,382	5	$48,360	$50	967	Installation Tools
Dart PowerTCP Emulation Tool	38,702	VC++	10%	3,870	14	$135,457	$499	271	Internet Communication Components
Xceed Winsock Library	79,998	VC++	10%	8,000	28	$279,993	$500	560	Internet Communication Components Network Communication Components
Data Dynamics ActiveCube	360,000	VC++	10%	36,000	126	$1,260,000	$599	2,104	On-Line Analytical Processing Components Print & Preview Components
Data Dynamics ActiveReports	690,000	VC++	10%	69,000	242	$2,415,000	$499	4,840	Reporting Components
Data Dynamics ActiveSizer	31,000	VC++	10%	3,100	11	$108,500	$99	1,096	Resizing Components
Desaware NT Services Toolkit	58,000	C++	10%	5,800	20	$203,000	$499	407	Security & Administration Components

Component	SLOC	Language	% Component Used	SLOC Avoided	Time Avoided (Person Months)	Cost Avoided	I Developer License	ROI (x:1)	Component Type
Sax ActiveX Comm Objects	27,000	VC++	10%	2,700	9	$94,500	$399	237	Serial Communication Components
Farpoint Spread	284,091	C, C++	10%	28,409	99	$994,319	$479	2,076	Spreadsheet Components
Data Dynamics ActiveBar	346,000	VC++	10%	34,600	121	$1,211,000	$479	2,528	Toolbar Components
Desaware ActiveX Gallimaufry	50,000	VB	10%	5,000	18	$175,000	$149	1,174	User Interface Collections
Sax ActiveX SmartUI	23,800	VB	10%	2,380	8	$83,300	$399	209	User Interface Components
Sax ActiveX Basic Engine	65,000	VC++	10%	6,500	23	$227,500	$399	570	VBA and Scripting Components
Desaware VersionStamper	214,000	C++, VB	10%	21,400	75	$749,000	$249	3,008	Version Control Components
Desaware SpyWorks Professional	529,000	C++, VB, .NET	10%	52,900	185	$1,851,500	$379	4,885	Windows API Components

Conclusion

When assessing the metrics for COTS components, one should also factor in the cost to evaluate and reuse this functionality. Based on his analysis of a number of published studies, Jeffrey Poulin in his book “Measuring Software Reuse” concludes that reusing software “takes 20% of the effort of new development”. Even given Poulin’s assumptions, the study shows that using COTS components represents significant ROI compared to the greater cost of creating the equivalent functionality from scratch.

The metrics supplied on COTS components indicate the potential cost and time avoidance relative to the application development effort, represented through the purchasing and deployment of mature, market proven, expert-built COTS components. Many of our SAVE-IT customers use the data supplied on COTS components to justify their build vs. buy decisions, and cost forecast reuse savings and time to market benefits.

By applying industry averages to assign value to COTS components, companies can gain valuable information on the benefits of reusing these components in addition to leveraging their internally created assets. This information can be deployed at anytime during the development cycle of an application - from project planning to final revisions - to determine if the use of COTS components for specific functionality will be a cost-effective solution to delivering better, faster, and cheaper solutions

We will be publishing the raw value data for COTS components collected on our public marketplace, this will include the SLOC behind component products and person months displaced per development language.

Component Vendor Data Courtesy of:

Dart Communications

Dart Communications was founded in 1994 to create quality components designed to support Internet communication development. Dart's development teams carefully design each component for ease-of-use and maximum range of effectiveness for both beginners and advanced developers alike. The results are tools that function in many development environments, such as .NET, Visual Basic, Visual C++, PowerBuilder, ASP, Delphi, C++ Builder and Office 97/XP.

Data Dynamics

Created in January 1996, Data Dynamics, Ltd., provides software tools and controls for application developers

using Microsoft design environments. The Company's primary product focus is on Data Analysis and Information Reporting. However, they also offer User Interface development products.

Desaware

Since its inception in 1991 as one of the first third party Visual Basic component vendors, Desaware has developed innovative software products to assist developers in their programming efforts. Based on experience going back to the days of Windows 1.0, the company understands the critical features needed by developers, sometimes presenting a solution before developers are even aware of the problem.

FarPoint

FarPoint Technologies, Inc. was founded in 1991 and is located next to Research Triangle Park (RTP) in Morrisville, North Carolina. The company develops and publishes professional components for Windows development. Their award-winning tools benefit corporations, software companies, and independent consultants around the world as a cost-effective solution for building distributed enterprise-wide applications for commercial or in-house use.

LEAD Technologies

LEAD Technologies has been producing imaging developer toolkits for the past 12 years. LEADTOOLS is designed to handle all imaging needs – from common loading, displaying and image processing, to complex and high performance imaging demands of the document, medical and Internet industries. LEADTOOLS can support projects requiring raster imaging, vector imaging or multimedia support, or even a combination of all three.

Sax.NET

Sax.NET, formerly Sax Software, specializes in developing components for communications, data compression, scripting, and user interfaces. Sax.NET was founded six years ago and is located in Eugene, Oregon.

Xceed Software Inc.

Xceed Software Inc. creates, markets and distributes software components for Microsoft Windows developers. Since its launch in 1994, Xceed has been devoted exclusively to the Microsoft platform. The company's very first product, Xceed Zip Compression Library, was built for Microsoft Visual Basic 3.0 and has been migrated to every Microsoft platform since, including ActiveX and the .NET Framework.

Footnotes:

[1] SAVE-IT consists of an enhanced and proven three-pronged commercial approach to establish the business drivers for reuse of software assets inside of an organization, and a scalable asset rich infrastructure to institutionalize reuse. The customizable solution may be packaged according to a customer's needs. It differentiates itself in the market on three proven levels comprising: SAVE-IT™ Process™, SAVE-IT™ Catalog™, and SAVE-IT™ Content™.

[2] Source: Cost Xpert Group, Estimating Software Costs, Author: William Roetzheim

[3] ComponentSource has used industry averages to assign value to COTS components. For more information on these please contact press@componentsource.com.

Revision History: First Published: August 27, 2002. May contain copyrighted data previous published and owned by ComponentSource.

About ComponentSource

ComponentSource is the world's largest marketplace and community for reusable software components for all platforms, and is first to market as a Software Asset Value Provider with the launch of SAVE-IT, Software Asset Value Engineering in Information Technology. With seven years' experience at the helm of the component industry, ComponentSource is able to transfer its experience in running the world's largest reuse center on its public marketplace to the corporate environment. SAVE-IT is a mature three-pronged approach that establishes effective enterprise-scale software reuse and is the backbone technology for the National Software Component Exchange and private sector customers worldwide. The respected barometer for the component industry, ComponentSource pioneered the open market for reusable software components in 1995, and continues to drive the market through its award-winning e-business model and groundbreaking work to establish the first widely accepted reusable component standard. A global e-business with customers in over 110 countries, ComponentSource is headquartered in Atlanta and has offices in Reading, England. For more information, please visit www.componentsource.com.