Commercial Exploitation of Components

Author: RichardVeryard
Version: March 2nd, 1999

richard@veryard.com
http://www.veryard.com

Preface

Purpose of document

This document is aimed at software suppliers and other software organizations wishing to sell software components.

It is relevant both to traditional software suppliers moving into CBD, as well as in-house software organizations wishing to gain commercial advantage from existing software assets.

It is also relevant to component brokers and other intermediaries, and to the builders of component repositories and support tools.

Status

Working draft. Guidelines and techniques not fully enumerated.

Questions and exercises

Q These questions are inserted for the reader to consider, or to discuss with colleagues and friends. In classroom situations, they may be suitable for short group discussions.

Acknowledgements

Introduction

One of the central ideas of CBD is that of a separation between a **service**, provided through an external interface, and the pieces of code that **implement** this service.

This separation is increasingly reflected in the software marketplace. Some software suppliers continue to focus on providing well-crafted pieces of code, while others have repositioned themselves to providing services.

This separation applies at any level of granularity. Thus even the supplier of a very small software component may be service-focused, while many package suppliers and application assemblers remain implementation-focused.

	Service focus	*Implementation focus*
Large-grained	Application package sold in terms of services.	Application package sold in the traditional manner.
Small-grained	Component sold as black-box service.	Component sold as white-box pattern.

There are some important potential strategic advantages for the supplier who is service-focused:

	Service focus	*Implementation focus*
Potential advantages	➢ Closer relationship with customer demand. Faster response to changing patterns of demand. ➢ Pricing based on use-value. Good profit margins for the right product. ➢ Opens up new charging options (e.g. pay per use).	➢ Closer relationship with technology.
Potential disadvantages	➢ Demands new capabilities throughout supplier organization. ➢ May not be able to respond as quickly to radical technological developments.	➢ Loss of control of customer relationship - dealing with customers through intermediaries. ➢ Commodity-based pricing. Profit margins squeezed by competitors, and eroded by intermediaries.

However, these strategic advantages are only available to a supplier with a high level of commitment to CBD by the enterprise as a whole, as well as a reasonable level of experience.

This document considers some of the implications of a CBD service focus on the business processes of a typical software supplier, and offers an outline solution.

Method

The approach outlined in this document is based on the SCIPIO method. It has the following components.

Market Analysis	What is the nature and structure of the demand for services that we can provide through software components? What specific requirements can we identify? How should we generalize these requirements? What are the dimensions of diversity of this demand? What are the design variables that will control market penetration? What is the potential value of these services to potential customers or users, and how can this be maximized?
Supply Architecture	How are the services to be configured and layered? How is the software that delivers the services to be configured and layered? How should we maintain a robust mapping between the service architecture and the software architecture? How much open access to these architectures do we want our customers to have?
Component Product Management	What is the “whole product” that is sold to customers? How will it be presented and packaged? What is required to support the service in addition to the software? What terms of sale? How will the customers pay for the acquisition, deployment or use of your components? How will you fix the price? What rights do customers have? Can they sell these services (perhaps bundled with other services) to their own customers? Can they modify the source code? What is your commercial target for this component? How many do you need to sell, to cover your costs, and to achieve the desired return?
Capability Development	What software development capabilities do you need to develop? What software product management capabilities do you need to develop? How do you need to change your business processes to support an entry into the component marketplace?

Market Analysis

Advanced modelling techniques are available for various aspects of market analysis, including:

- To analyse the structure of demand in a market, where the market is regarded as a complex ecology.
- To analyse the structure of requirements.
- To quantify the demand, and the distribution of demand.

Understanding the structure of demand.

Key Issues

- What is the nature and structure of the demand for services that we can provide through software components?

Understanding the structure and distribution of requirements.

Key Issues

- What specific requirements can we identify?
- How should we generalize these requirements?
- What are the dimensions of diversity of this demand? What are the design variables that will control market penetration?

Guidelines

One of the important concepts of component requirements is **requisite variety**. A component or service with too much variety is not manageable; a component or service with too little variety is not likely to achieve a satisfactory market penetration.

There may be variety in both functional and “non-functional” requirements. Some users may want high performance and high reliability, and be prepared to pay a premium price for component services with these qualities. A user with only a thousand transactions per month may not want to pay for a component service that is designed to handle a million transactions an hour.

Quantifying demand, and the distribution of demand

Key Issues

- What is the overall size of the market?
- How much demand is there for any given permutation of features and qualities?

Quantifying value

Key Issues

- What is the potential value of these services to potential customers or users, and how can this be maximized?

How much analysis?

One of the key questions for any software supplier is: how much market analysis do I have to do, and do I need to use formal methods? We can identify three cases.

Implicit	**Explicit**	
Informal		**Formal**
Private intuition	*Shared intuition*	*Sophisticated analysis*

At one extreme, an understanding of the market is implicit, taken for granted. Some of the people involved may have some private and intuitive notions of market structure and forces, but these are not openly discussed.

Q Is this ever a valid approach? How much experience of a market does a person need to make reliable intuitive judgements?

At the other extreme, the market analysis is carried out formally, using advanced market modelling and market research techniques.

Q Is this always a valid approach? How much capability and prior experience does an organization need to be able to make good use of sophisticated research?

In between these two extremes, there is a shared market analysis process. Market analysis is still largely intuitive rather than formal, but it can be challenged and refined in peer-group discussion. Where this process reveals specific areas of uncertainty or risk, these can be explored more formally.

For most organizations new to this game, it is wise to start in the middle, drawing on external expertise where necessary to support a fairly simple but complete market analysis.

As the organization gains experience in the commercial exploitation of components, it can start to use more advanced modelling techniques to produce more precise answers.

Supply Architecture

My friend David, who is an expert on software components, recently complained about the maintenance on his expensive German automobile. He wanted the garage to fix a faulty rear light. It turned out that the light was part of a larger component and he needed to have the entire rear light cluster replaced, costing several hundred pounds.

He asked who made the decision to set the unit of replacement at the level of the rear light cluster. Was it a production engineering decision? Surely the marketing role would not have misrepresented customer requirements so badly?

Sadly for us consumers, the key question for many marketing departments is not

> *(A) which component architecture would be most convenient for most of our customers?*

but

> *(B) does this component architecture make any difference to the likely volume of sales?*

In other words, does the car-maker care whether it costs you more to fix your rear light? How much money do they make from selling spare parts?

(Perhaps the most skilled marketing departments are those that manage to put an A-style presentational spin on a B-style decision. In other words, they can persuade the customers - and even the industry analysts - that they put the customers' interests above their own. And then there's Microsoft.)

Another industrial analogy is the mobile phone. Most of us have good-looking phones, with lots of features, but with an almost unusable menu structure, with strange interactions between the features of the phone and the features of the connection service (I have not succeeded in sending a fax from my phone yet) and no upgrade path (I cannot use some of the accessories that are available for my type of phone, because my phone has an earlier version of the software). As long as we consumers tolerate this, the manufacturers have no incentive to improve these aspects of the product design.

The relevance of this for software components should be obvious. Any software vendor (or consortium) that designs a component architecture that is not driven by their own business advantage is naive and therefore strategically vulnerable.

Configuring services to satisfy market demand.

Key Issues

- How are the services to be configured and layered?

Configuring software assets to deliver services.

Key Issues

- How is the software that delivers the services to be configured and layered?
- How should we maintain a robust mapping between the service architecture and the software architecture?

Controlling access to architecture.

Key Issues

- How much open access to these architectures do we want our customers or third parties to have?
- How (to what extent) do you intend to control customization and modification?
- Do you want to provide public access to internal interfaces? Which ones, to what level of granularity? Do you want to provide public access to design or code?

Understanding business implications of architectural decisions.

Key Issues

- How does this architecture affect customer perceived value?
- How does this architecture affect our production, marketing and support costs?
- How robust is this architecture in the face of likely competitor / industry behaviour and trends?

Component Product Management

Product Planning

Key Issues

- What is your commercial target for this component? How many do you need to sell, to cover your costs, and to achieve the desired return?

Product Scope and Variety

Key Issues

- What is the "whole product" that is sold to customers? How will it be presented and packaged? What is required to support the service in addition to the software?

Pricing / Packaging

Key Issues

- How much are your components worth to your customers?
- What terms of sale? How will the customers pay for the acquisition, deployment or use of your components? How will you fix the price?

Guidelines

Charging options include:

- One-off payment upfront.
- Free use of software for trial period, followed by one-off payment.
- Free use of software for development. Payment per installation.
- Payment per time period.
- Payment per use.
- End-user product distributed free. Payment for developer or server products.
- Use is free, funded by advertising.

Q Think of examples of popular software and related services that adopt any of these charging mechanisms.

Q What other charging mechanisms can you think of?

Packaging / Publication

Key Issues

- What are the options for delivery and operation?
- What rights do customers have? Can they sell these services (perhaps bundled with other services) to their own customers? Can they modify the source code?
- Is there a demand for a source code escrow service?
- How frequently do you intend to release new versions & upgrades?

Guidelines

The supplier should probably aim to control the adaptation of components - but only if both of the following conditions hold:

- You have the capability and resources to perform and support any required adaptation itself. (This depends on a careful assessment of your processes.)
- Customers accept that you will perform all adaptation. (This depends on a careful analysis of market forces.)

Organization Learning

As described in this document, the "ideal" way to sell components is to define a set of services that meets a defined market/customer demand, and then design the best way to deliver these services as a set of components.

However, the "ideal" way is not suitable for all situations.

Many software suppliers, even fairly sophisticated ones, are not yet ready for this. They are not yet prepared to integrate CBD fully into their business strategy, although they are at least starting to introduce CBD into their software architectures and development processes.

If this is true for your organization, then you will need to find adhoc answers to the key questions, and accept that these answers will need to change over time, as your organization starts to get more fluent with CBD in sales and marketing, and also in customer/product support.

Product Innovation

Key Issues

- What software development capabilities do you need to develop?

Process Innovation

Key Issues

- What software product management capabilities do you need to develop?

Planning Innovation

Key Issues

- How do components fit your business strategy?
- How do you need to change your business processes to support an entry into the component marketplace?
- What is the level of component awareness & capability
  - Within your organization?
  - Among your customers?
  - Among key third parties?

Checklist of Considerations

Here are some preliminary considerations, for new entrants to component marketplace.

- What is your process for determining the requirements for components? Do you start with a single application requirements or customer request, and then generalize? Or do you start with some kind of domain analysis?

- Do you try to estimate how many customers will buy a given component, and how many times will each customer use the component? How are such estimates obtained, and what use do you make of them?

- What is your current level of experience / expertise with component-based development? What are your plans for going to higher levels of expertise?

- What level of experience / expertise / understanding / awareness do your customers have with component-based development? What about other players in your marketplace, such as competitors, consultancies and system integrators?

- What is your business strategy, and how does the sale of components support this strategy? What are your business relationships with its customers, and how is CBD expected to affect these relationships?

References

Other SCIPIO white papers are available via the SCIPIO website: http://www.scipio.org/. and the Veryard Projects website: http://www.veryard.com/.