

DSDM and Component-Based Development

Table of Contents

1	INTRODUCTION.....	3
1.1	AUDIENCE.....	3
1.2	CONTRIBUTORS.....	3
1.3	STRUCTURE.....	4
2	SCOPE AND EXCLUSIONS.....	5
2.1	SCOPE.....	5
2.2	EXCLUSIONS.....	5
3	BASIC PRINCIPLES OF CBD	6
3.1	SOME BASIC QUESTIONS ANSWERED.....	6
3.2	THE IMPORTANCE OF ARCHITECTURE.....	7
3.3	COMPONENT GRANULARITY	7
4	CBD WITH DSDM	9
5	REVOLUTIONARY CBD	11
5.1	OVERVIEW	11
5.2	BENEFITS OF REVOLUTIONARY CBD	11
5.3	ISSUES.....	12
5.4	STEPS IN THE REVOLUTIONARY APPROACH.....	12
5.5	DSDM SUITABILITY RISK LIST	15
6	EVOLUTIONARY CBD	17
6.1	OVERVIEW	17
6.2	BENEFITS OF THE EVOLUTIONARY APPROACH	17
6.3	ISSUES.....	18
6.4	STEPS IN THE EVOLUTIONARY APPROACH.....	18
6.5	DSDM SUITABILITY RISK LIST	20
7	ASSEMBLY CBD	22
7.1	OVERVIEW	22
7.2	BENEFITS OF THE ASSEMBLY APPROACH	23
7.3	ISSUES.....	24
7.4	STEPS IN THE ASSEMBLY APPROACH.....	24
7.5	DSDM SUITABILITY RISK LIST	26
8	PROCESS AND PRODUCTS.....	28
9	ROLES AND RESPONSIBILITIES	32
9.1	INTRODUCTION	32
9.2	DSDM/CBD ROLE DEFINITIONS.....	33
9.3	EXTRA SPECIALIST DSDM CBD ROLES	37

10	ROLES AND CBD PROJECT TYPES.....	38
10.1	REVOLUTION	38
10.2	EVOLUTION	39
10.3	ASSEMBLY	41
11	FUNDING MODELS FOR CBD	43
11.1	CBD FUNDING CHALLENGES	43
11.2	CORPORATE V. PROJECT FUNDING	43
11.3	FUNDING THE CBD SCENARIOS	43
11.4	CBD COSTS AND BENEFITS.....	44
12	BIBLIOGRAPHY	48
	APPENDIX A COMPONENT DEFINITIONS	49

1 Introduction

The objective of this white paper is to allow senior management to clarify their thinking about their strategy for CBD as follows:

- To outline the principles and explain the relevance of CBD with respect to DSDM
- To describe the different ways in which CBD might be applied (i.e. scenarios).

To facilitate the choice of scenario:

- **Revolutionary Approach:** where corporate modelling is followed by build of all components and then solution assembly.
- **Evolutionary Approach:** where scope partitioning is followed by solution development and then the harvesting of components.
- **Assembly Approach:** where components are identified, purchased and assembled into solutions.

The paper details additional issues such as funding models and new roles. Once this strategy has been decided it should be possible to follow the standard DSDM manual supported by the additional guidance contained in this white paper.

1.1 Audience

The paper is intended to be read by anyone concerned with applying DSDM on CBD projects and, conversely anyone wanting to apply the principles of CBD in a DSDM context. This includes the Project Manager, Team Leader, and Visionary roles together with any member of a DSDM project governing body such as a project board or steering committee.

The paper assumes a certain level of knowledge of components and CBD; see Section 3 for basic principles, Section 12 for references and Appendix A for definitions.

1.2 Contributors

This White Paper was put together by a Task Group reporting to the UK Technical Work Group:

Paul Allen	Sterling Software (Chair)
Nick Boxford-White	Maritz
Mark Edwards	Willis
Howard Lewis	Morgan Stanley
Christine Robertson	Clerical Medical Investment Group (Co Chair)
Kevin Trembath	Xansa
Paul Turner	Parity Training
Steve Ash	OO Training and Consultancy

This White Paper was updated to DSDM Version 4.2 by Kees Peeters and Rik Jan van Hulst.

1.3 Structure

Section 2 lists the scope and exclusions of the paper.

Section 3 aims to clarify what CBD is and discusses some of its implications.

Section 4 highlights the important activities which must surround the DSDM activities for CBD to be effective.

Sections 5 to 7 discuss the three CBD scenarios: revolution, evolution and assembly. These sections show the processes to follow in each scenario and covers the risks and benefits of enacting each scenario. Neither the products nor the roles are discussed in these sections.

The products produced by DSDM/CBD work are listed in Section 8.

Section 9 shows the roles and responsibilities which are added to the DSDM roles and responsibilities and shows which roles should be present for each of the three CBD scenarios.

Section 10 considers funding models for the three CBD scenarios and some ways of justifying the costs.

2 Scope and Exclusions

2.1 *Scope*

The scope of this white paper is to facilitate decision-making on the effects that CBD can have on DSDM projects.

2.2 *Exclusions*

The following are excluded from the white paper:

- Guidance on CBD methodologies, including “how to....” advice.
- Considerations of software packages, visual modelling tools and core component technologies and standards.
- Changes to the DSDM lifecycle framework and principles. The framework is unchanged but additional roles and activities are proposed. DSDM's underlying principles all apply however the application of the principles in CBD is discussed where necessary.

3 Basic Principles of CBD

3.1 Some Basic Questions Answered

What do we mean by “component”?

Different definitions of component (see Appendix A) share some common themes that are relevant in the context of this paper and that are summarised in this paragraph. A component’s functionality and data is encapsulated; that is, only exposed in terms of services provided via defined published interfaces written to an interaction standard. An interface describes only *what* is offered, regardless of implementation design or executable. Providing the interface remains consistent, a component can be replaced with another having different implementation code, without disturbing clients. This feature allows components to be implemented and deployed using different technologies, while retaining the same interface specification. Once delivered as an executable, components are platform specific. However, the component technology masks the actual location from the requesting client, making it appear local to the requesting application.

What do we mean by CBD?

CBD refers to the architectures, procedures, techniques, and organisational guidelines associated with building and reusing components across the software marketplace. Your view of CBD depends on your position in the software marketplace as illustrated in Figure 1. At one extreme component suppliers are concerned with quality engineering of generic components to rigorously specified interfaces: this can be characterised as an architectural or *revolutionary* approach. At the other extreme an end-user organisation may focus on fast-track assembly of solutions from pre-built components: this is the *assembly* approach. In between are various blends of the two approaches. In particular, there will be many end-user organisations who want to minimise risk by developing components “as you go”, generalising components as “spin-off” from business solutions development: this is the *evolutionary* approach.

Why is this important?

DSDM and CBD are inter-dependent processes. DSDM depends on CBD for a supply of pluggable components. CBD depends on DSDM to develop software in timely fashion, prior to certification as components. DSDM emphasises a business-driven development culture that is fast becoming characteristic of successful software development. Components must be seen as useful from a business point of view. Any project that uses CBD within the context of DSDM must keep the project focussed on business goals and ensure that business members are comfortable with the principles.

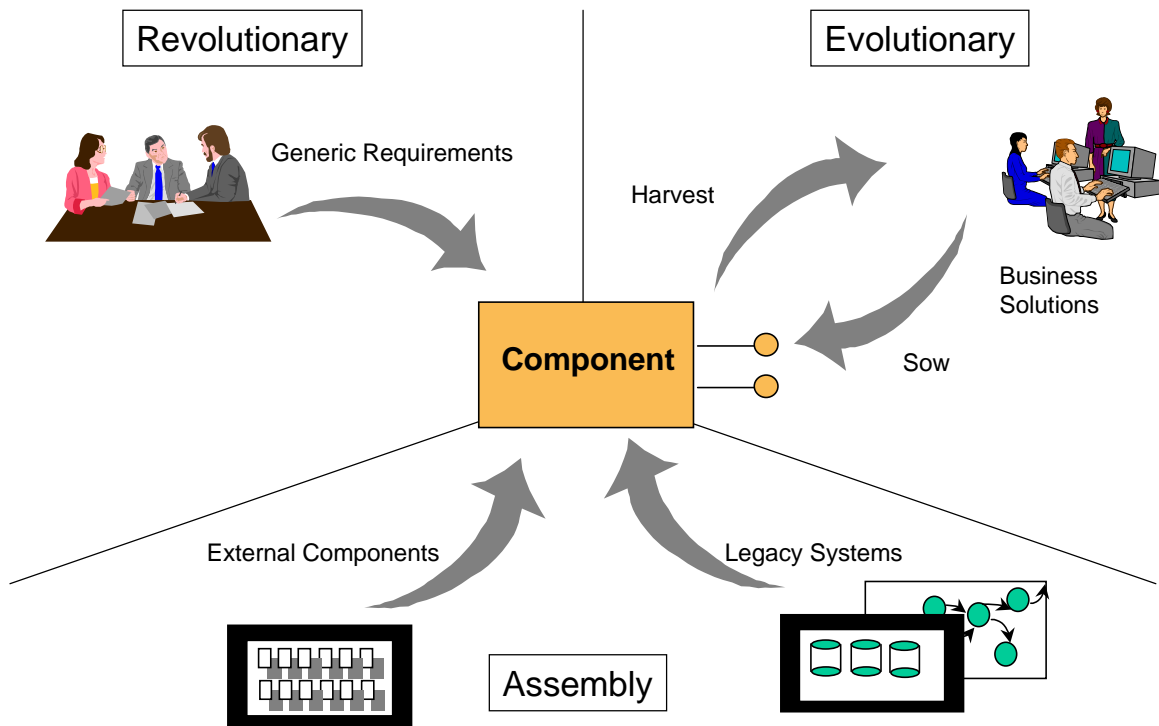


Figure 1: CBD Scenarios

3.2 The Importance of Architecture

A component-based approach has the potential to remove the tight coupling of an application's parts and ease reuse of parts across different applications. The great promise is that application "islands" disappear: functionality and data are packaged as components, which are configured in network fashion to meet business needs. The assumption here is that components are well designed. Good design in turn depends on evolution of an effective architecture.

An effective architecture must provide an overall structure and set of rules for managing the scale and complexity inherent in enterprise software development. It must help achieve software interoperability, adaptability and consistency of design. The most successful business components are those that can be "rewired" in many configurations in effective response to business change.

3.3 Component Granularity

Reuse of technical components, from dragging and dropping of GUI components to reuse of database connectivity modules is now pervasive. Similarly, many organisations have bought heavily into large-scale package solutions in the later half of the 1990's. Business components generally fall between these two extremes in terms of granularity as illustrated in Figure 2. Business components may themselves vary in granularity from lower level address formatting to much higher level business frameworks that allow the developer to extend higher level concepts such as Product or Customer.

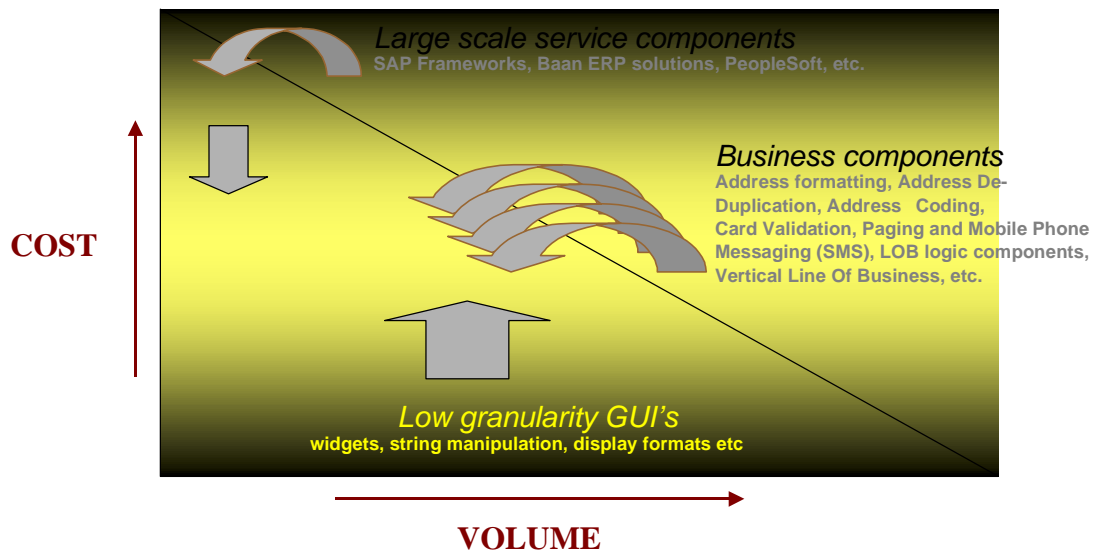


Figure 2: Component Granularity in Industry Context

Business components have been slow to gain acceptance, partly because of a lack of available trusted business components and partly because of the notorious difficulties in reaching consensus on the business problems that DSDM is particularly suited to address. We are also seeing the emergence of component frameworks and libraries that form another input to DSDM projects, always concerned with achieving good business results.

4 CBD with DSDM

The DSDM Lifecycle

CBD involves applying infrastructure activities, described below, to the basic DSDM lifecycle as illustrated in Figure 3. This consolidates common themes that run through each scenario. A benefit of this lifecycle framework is that it leaves the existing DSDM lifecycle unaltered, simply wrapping the new CBD infrastructure activities around it. Planning and architecture provide an evolving context for DSDM projects in a CBD setting. Component management provides the search, catalogue, certification and publication services that operate on the repository of components.

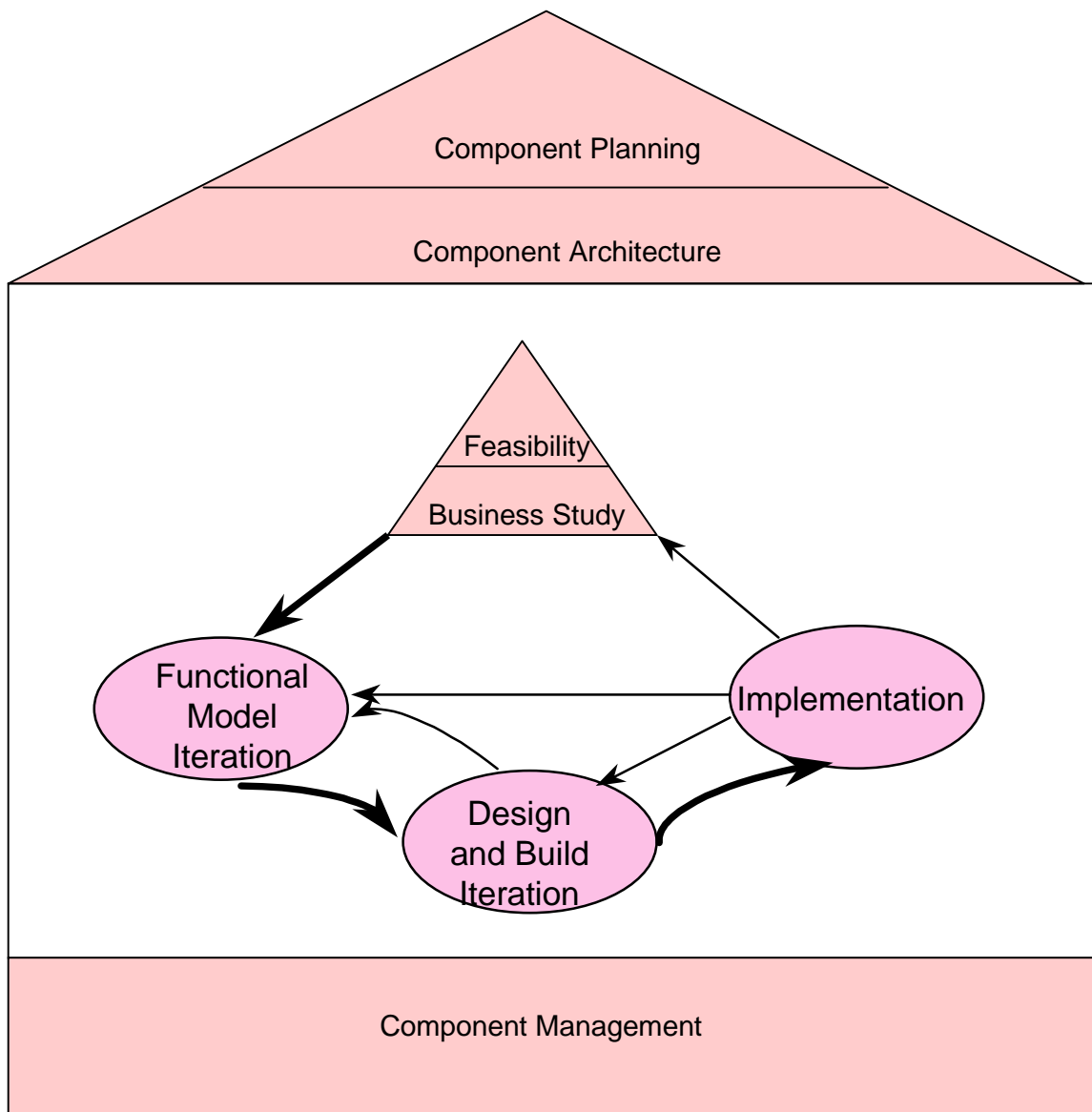


Figure 3: DSDM in the Context of CBD

Component Planning

Component planning must cover the overall business case for CBD. This includes both provisioning strategy and quality planning. The provisioning strategy includes policy on build, reuse or buy decisions and is heavily influenced by the particular route (revolution, evolution or assembly) that the organisation has chosen as described in sections 5 to 7. Provisioning strategy also influences the choice of funding model described in section 11.

Component Architecture

A component architecture provides an overview of the software parts from which the envisaged software will be built, and describe how those parts interrelate. Both business and technical architectures are included (see Section 5). The software parts may be encapsulated components, non-component assemblies, software packages and legacy systems. The component architecture provides a "big-picture" for projects to work to. This is particularly important where incremental development is undertaken or where parallel work is performed by relatively independent teams.

Component Management

Class browsers that assist in the development of OO systems have been widely available for some years. One of the obstacles to component reuse has been the absence of effective facilities for cataloguing and retrieving components in a business-oriented way. Some component management tools now provide component catalogues that hold component information within a repository and provide the ability to browse, install and register the components in harmony with model-driven approaches.

5 Revolutionary CBD

5.1 Overview

Background

The purpose of this section is to explain the revolutionary approach to CBD where a strategic view has been taken to building and assembling components that meet the requirements of business processes.

The aim of a revolutionary CBD approach is to significantly improve systems flexibility and capability to match changing business processes, and reduce the cost and effort required developing and maintaining future systems.

Approach

The revolutionary approach requires a common understanding of current business needs and future direction, and up-front investment to develop a corporate model for the business. It is imperative that a view of the future is taken both in business and technology terms.

Given that Revolutionary CBD has an enterprise focus, it takes an organisation-wide view of business processes and is therefore, primarily interested in end-to-end business processes. End-to-end processes go right across the business units and are typically triggered by external sources (e.g., a customer requesting a service), as illustrated in Figure 4.

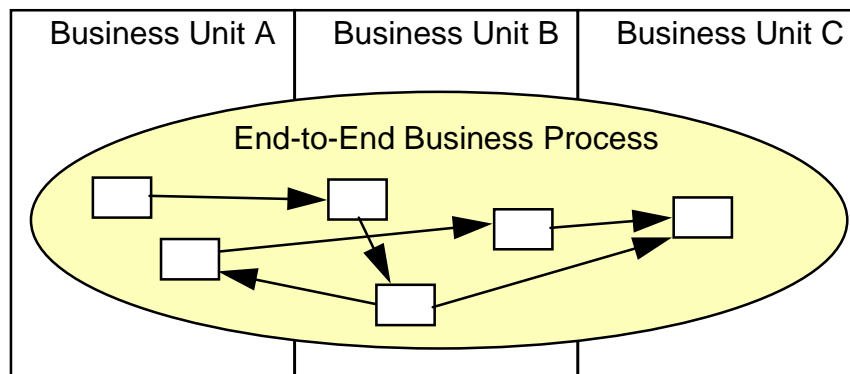


Figure 4: End-to-end Business Processes

Once most of the components are available for re-use, then the amount of strategic budget that is required is substantially reduced. Moving to a combined revolutionary and evolutionary approach should be considered once the component build is stable and the experience has been gained.

5.2 Benefits of Revolutionary CBD

In this approach, business requirements are analysed and a business model is created whose aim is to define as much of the business as possible as reusable components.

The objective is to ensure that common business data and behaviour that is likely to be required by multiple applications is specified (and subsequently developed) only once. Applications both contribute to and benefit from a growing component inventory. The component inventory thus becomes a corporate asset.

A strategic approach enables the overheads of the component build process and the associated development costs to be removed from the project estimates. It helps to remove the conflict of priorities and the different objectives between project and architectures and allows either effort to focus on best meeting its objectives without having to make 'risky compromises' to accommodate the other.

5.3 Issues

There are some key areas that are affected by adopting a revolutionary CBD approach:

- **Cost** Revolutionary CBD is cost-efficient in the long term as it optimises business investment by ensuring that there is no duplication and promotes a reuse culture. This approach means that, in the short term, development time-scales increase. In the long term, when the component build is substantially complete, delivery times should be reduced.
- **Configuration Management** The role of Configuration Management will expand due to the complex interoperability requirements that CBD imposes. There is an increase in the likelihood of incompatibilities between software elements in the production environment.
- **Visionaries and Users** It is essential that the business is fully bought in and committed to the creation of the Business Architecture Model. As in all DSDM-based work, it is the business who will provide the detailed knowledge required. As always, the visionaries and users identified must be of the right level and have the authority to make decisions and must also have the relevant business experience, expert knowledge and vision.

5.4 Steps in the revolutionary approach

Figure 5 shows the revolutionary approach using the DSDM lifecycle to build and extend components.

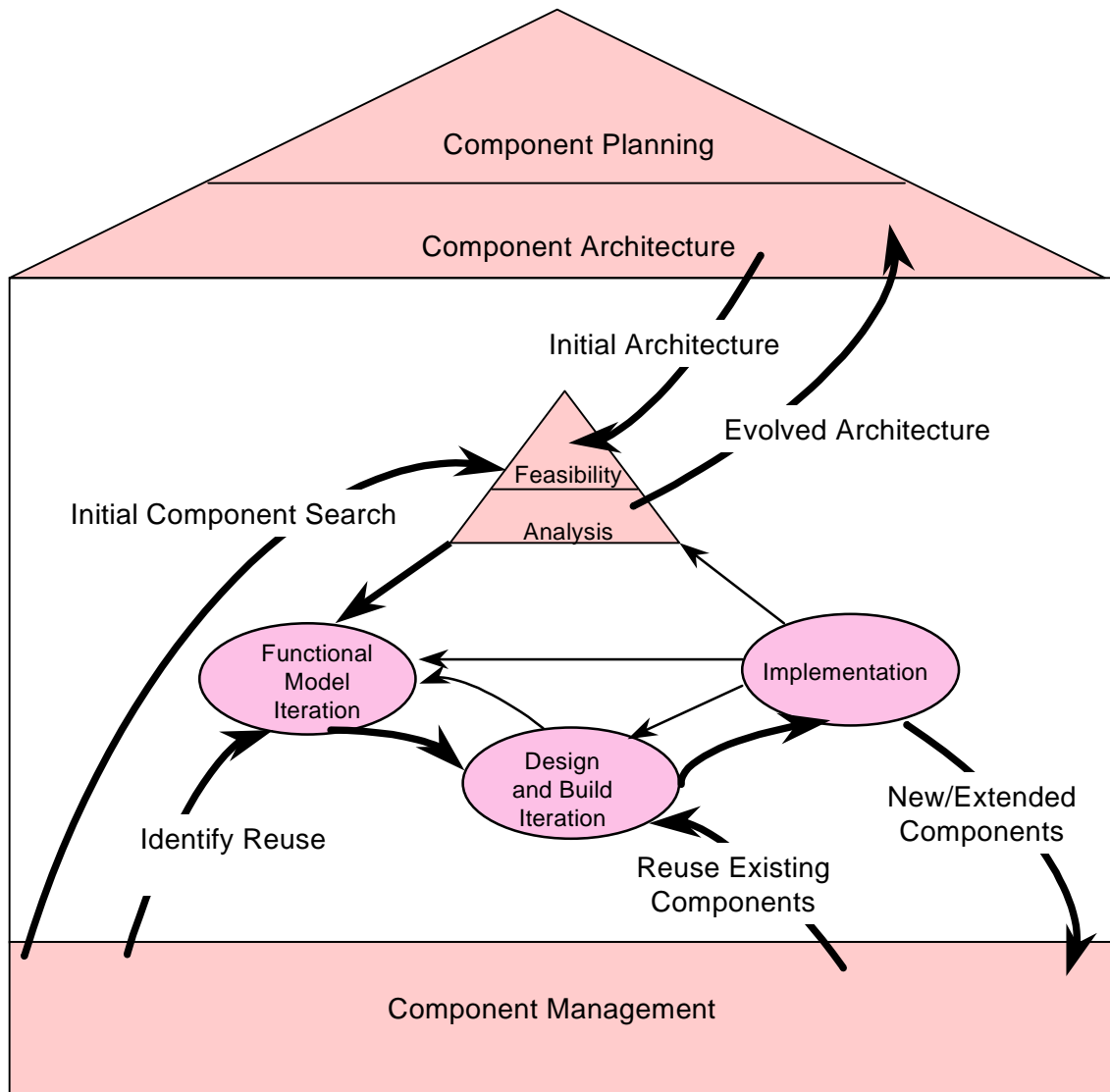


Figure 5: Revolutionary Process Overview

The revolutionary approach should perform the following steps.

1) **Create Business Architecture**

An initial architecture should be established which includes a high-level logical view of the business by subject area and suits the size and reuse priority of the Business. The Business Architecture Model consists of a domain model showing major areas of the Business with key dependencies. The architectural model defines the basic infrastructure, incorporates all aspects of the business and focuses on stable long-term assets. A collaborative and co-operative approach is essential at this stage between the Visionaries and IT. During this stage, existing systems should be evaluated and the Business Architecture Model validated and expanded to include any systems where the business functionality requirements are met on a sound technical system.

2) **Define Technical Architecture**

The Technical Architecture models the technological framework for building information systems (including tools, languages, middleware, standards, protocols, third-party components, etc.). The main focus of the technical architecture is on construction. The architecture needs to continuously evolve and the following should be considered during its design:

- Use of open technologies, especially those that conform to widely accepted standards, will give greater control over design choices.
- Technologies that are demonstrably more productive, not just in their initial use, but also for long-term maintenance purposes.
- The need to deal with legacy systems. The technologies chosen should provide a practical migration path.

3) Identify Components

For each component identify whether the requirement is to:

- **Buy/Build** where there is little coverage of the required functionality and what does exist is not technically sound;
- **Wrap** where all required (and foreseen) business functionality is currently provided, but where the technology assessment does not fit with the technical infrastructure;
- **Enhance** where the technical base is sound, but further business functionality is required;
- **Use 'As Is'** where the business functionality requirements are met on a sound technical system.

4) Design Components

The Business Architecture Model allows logical components to be identified which provide “first cut” physical components. Agree and fully design the preferred solution.

5) Build Components

Build, buy and/or enhance components so that they provide the agreed functionality as well as supporting all existing functionality. The build and implementation of components should be substantially driven by the dependencies that exist between components in terms of the services they request from, and provide to, one another. Testing should be integrated throughout the component build.

6) Integrate Components

Integrate components with front-end systems/GUIs to form solution. Testing should be ongoing throughout the integration stage.

7) Implement

Deliver and make available to DSDM teams a tested system solution in parallel with any required business changes. Implement first the components, or the elements of components, which provide services to many other components. This approach will enable these services to be reused as early as possible, and would provide a basis or

framework to which other components could be added in order to provide complete services of value to the business.

5.5 *DSDM Suitability Risk List*

The major risk to the feasibility of achieving successful delivery of the components is one of manageability. Table 1 contains a modified DSDM Suitability Risk List for use when assessing the risks of revolutionary CBD. In Table 1, “project” refers to the development of a component.

Suitability Factor	Y/N	Comment
Is there commitment to user involvement?		User involvement throughout the lifecycle is essential and especially when defining the Business Architecture where Visionaries are crucial.
Will the project team have the appropriate skills?		High risk: There will probably be little knowledge of CBD at the outset.
Is there clear ownership?		Conflicts between immediate needs of the business and mid/long term views of the architects may be an ownership issue.
Will the approach call for the use of technology that has not been used before?		High risk: In many cases the introduction of CBD will have an effect on the technical infrastructure. There is a strong likelihood of new technologies being introduced.
Has the project a fixed time-scale and /or budget?		The scale of the project will make it difficult to provide accurate estimates.
Could the project efficiently re-use (e.g. software components, business process and experience) from previous developments?		Low risk: The identification of re-use is a step in defining the Business Architecture.
Are the requirements not too detailed and fixed?		The potential impact, during the migration period, of unexpected external factors such as legislation, market changes and technological innovation should be considered.
Is there any previous experience in this type of project?		The Revolutionary CBD approach will require a degree of cultural changes for both IS and the business.
Will the project conform to architectural guidelines		Establishing both Business and Technical Infrastructures at the outset are key in

Suitability Factor	Y/N	Comment
(technical and business)?		Revolutionary CBD.
Are the component teams empowered to make decisions?		The teams may need to be over-ridden by the architects to ensure compliance with the “big picture”. This can be mitigated by communication of the “big picture” at the start of the exercise.

Table 1: DSDM Suitability Risk List for Revolutionary CBD

6 Evolutionary CBD

6.1 Overview

Background

The Evolutionary approach to CBD sets out to incrementally deliver both business applications and reusable components in parallel.

The approach takes as input a technical architecture and business vision. It partitions a business domain into manageable increments. Each increment delivers business benefit and contributes to the evolution of a fit for purpose corporate component model.

Risks of combining this approach with DSDM are centred on the conflicts between the mid/long term view of the project architects and the increment view of the development team. These risks are far outweighed by synergy created by the iterative and incremental nature of both DSDM and Evolutionary CBD.

Approach

The approach relies on a good understanding of the business domain and a business vision. This is required to allow partitioning of the domain into coherent functional increments and to ensure future directions are allowed for. The approach does not mandate up-front corporate component modelling. However, it does rely on the provision of a technical architecture. This means that decisions on interoperability standards have been made, middleware products chosen and the necessary application architecture frameworks added. This infrastructure should be provided to the application development teams. The architecture should act as an enabler of system delivery.

The provision of architecture will support black box modelling in the Business Study when the early analysis and high-level design are carried out. The only concern of the modellers is the business domain; the underlying plumbing that supports the system must be a given.

The approach relies on incremental delivery. Projects are partitioned into manageable increments. Vertical partitioning should be used (i.e. end to end through the architectural layers) so that business benefit is delivered by each increment. As in all DSDM projects, the need for good configuration management increases with incremental delivery.

The approach relies on iterative development within an increment and between increments to evolve components to fit the business need. Treating the components as black boxes with defined services easily supports iteration. Iteration of an interface is not allowed unless backward compatibility is assured.

The approach mandates the use of timeboxing, in combination with iterative development of fit for purpose components.

6.2 Benefits of the evolutionary approach

- Early delivery of business benefit. The combination of component development and delivery of business solutions from the first increment results in early business

benefit. As a secondary benefit this approach results in the generation of team and project momentum.

- Optimisation of “fit for purpose” components. Each increment follows the MoSCoW prioritisation and timeboxed development principles of DSDM. This avoids over-engineering both the requirements and the solution: in turn, the loss of momentum caused by “analysis paralysis” is avoided.
- Projects are of a manageable size. The incremental delivery strategy mandated in the evolutionary approach results in team sizes and delivery scope which are well suited to DSDM.
- Changes in the business domain are easily allowed for. Incremental delivery allows for “just in time” component analysis and design. In this way, impacts on the component model from changes in the business domain are minimised. The components delivered at the end of the final increment represent a more accurate snapshot of the business domain than would have been obtained at the start of the first increment.

6.3 Issues

- All changes are reversible. The configuration management challenges facing the evolutionary approach are significant
- Requirements must be baselined at a high level. The level of detail of the requirements must be low enough to allow “black box” component modelling. This may mean that the baseline is set at a level that limits the margin for creativity of the development team in providing system solutions within the high level requirements framework.

6.4 Steps in the evolutionary approach

Figure 6 shows the evolutionary approach using the DSDM lifecycle to build and extend components.

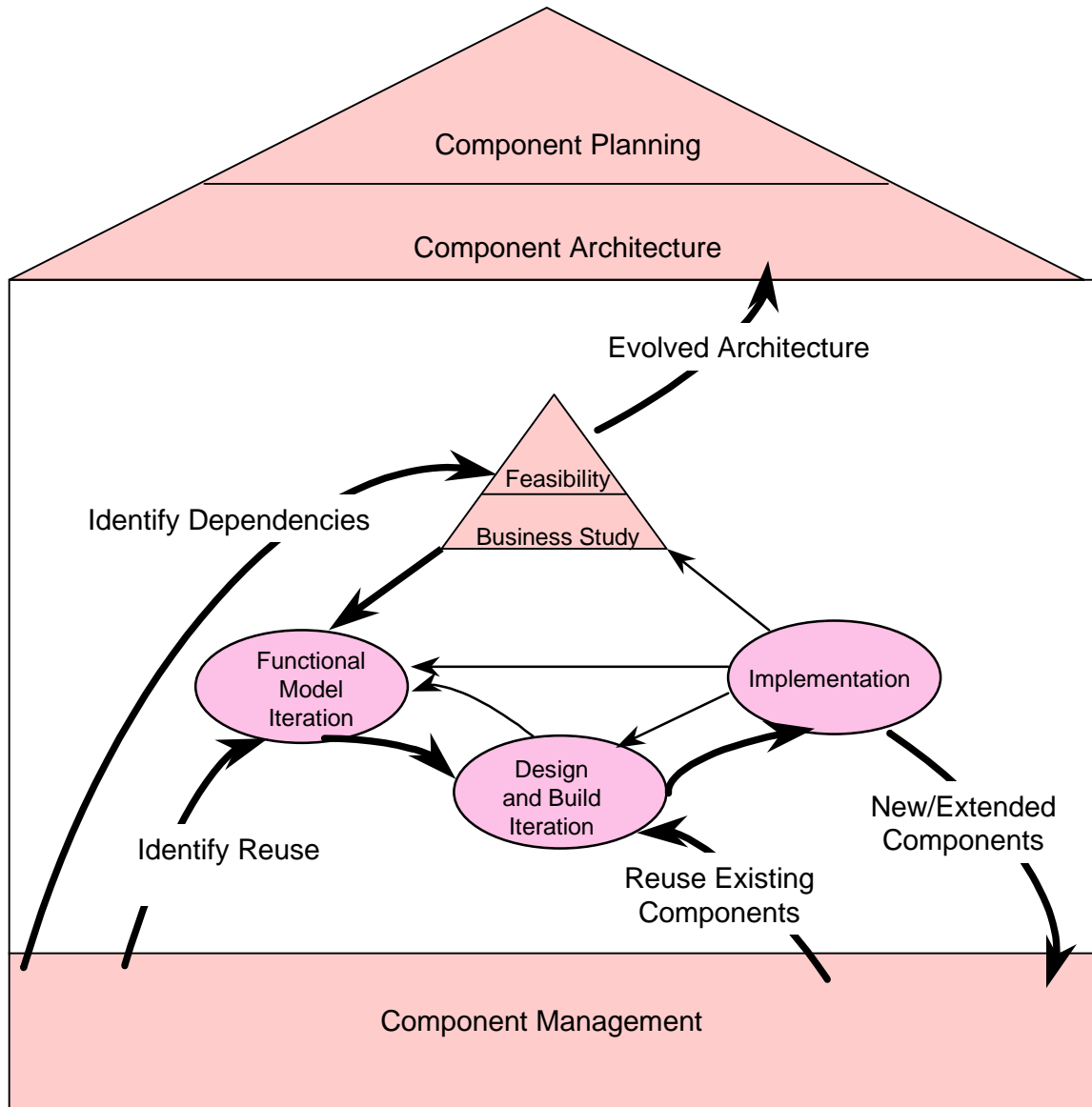


Figure 6: Evolutionary Process Overview

The evolutionary approach should perform the following steps.

- Business Study:
 - Identify and analyse business requirement. Partition the business domain into manageable, coherent business increments. Scheduling of these increments creates a programme/project relationship.
 - Define the technical architecture, standards and configuration management procedures.
 - Model the first increment. This will define the domain component model for the agreed scope. Publish the first iteration domain of the component model.
- FMI and DBI (first increment)

- Iteratively build the first increment. This will create the physical components for use in the increment and for reuse by other increments.
- Implementation
 - Certify and publish the resulting components.
 - Model the subsequent increments. Use the published component model as a source of domain level reuse. Extend or model new components as required (under the censorship of the business architect).
- Further increments
 - Build the subsequent increments. This will reuse, extend or create components for future use. Certify and publish the resulting components.
 - Evolve architecture to reflect component specifications and dependencies.
 - Continue until no new components are created or extensions added. At this point the domain model will be complete and only component assembly is required.

6.5 DSDM Suitability Risk List

Table 2 contains a modified DSDM Suitability Risk List for use when assessing the risks of evolutionary CBD.

Suitability Factor	Y/N	Comment
Is there commitment to user involvement?		User involvement throughout the lifecycle is essential to converge on a fit for purpose component implementation.
Are the teams empowered to make decisions?		The teams may need to be over-ridden by the architects to ensure compliance with the “big picture”. This can be mitigated by communication of the “big picture” at the start of the exercise.
Can the solution be developed in increments		The iterative and incremental nature of the process emphasises the importance of this factor.
Can the organisation accommodate the frequent delivery of increments		The iterative and incremental nature of the process emphasises the importance of this factor.
Will the individual teams consist of less than 6 people?		As always, each sub-team will contain 6 people but the entire team will need to be larger than this.
Can the requirements be prioritised		There is an added complexity of mapping functional requirements to components and operations.

Suitability Factor	Y/N	Comment
Is there commitment to user involvement?		User involvement throughout the lifecycle is essential to converge on a fit for purpose component implementation.
Is there clear ownership?		Conflicts between immediate needs of the business and mid/long term views of the architects may be an ownership issue.

Table 2: DSDM Suitability Risk List for Evolutionary CBD

7 Assembly CBD

7.1 Overview

Background

This scenario describes the situation where DSDM is the chosen vehicle for assembling a system from components obtained from an internal library or the open market. This does not include cases where a full package solution is to be implemented, as this is dealt with by another DSDM White Paper. The emphasis in a full package solution is on purchasing software inclusive of a standard user interface and adapting work practices to fit. In contrast, the emphasis with components is to assemble from components, which comprise of the business and data layers, with the responsibility for production of the user interface lying with the developer. In addition, the focus here is on reuse of sub-assemblies and business components rather than technical components, although these may also be used as in any DSDM project.

Approach

It should be appreciated that the range and volume of components available to buy for such use is still growing and this scenario is likely to become more common over time. The world of pre-built components will continue to extend beyond low granularity GUI components. However, the acute scarcity of resources in IT means that any functionality that can be reused is a very valuable commodity. When searching for appropriate components for use in the assembly process, granularity should be a major consideration. Where suitable components are identified, but not used, this decision must be justified to the DSDM Project Manager and Technical Co-ordinator.

During the DSDM project, the team designing the solution to a business requirement will identify parts of their business model, which they believe can be implemented using ready-built software components. This is followed by a search via a number of component catalogues to identify a suitable candidate. This is then used as part of the assembly process to provide a prototype for evaluation by users in the normal way. It may have been mandated up-front that components must be bought from a specific source.

In some cases candidate components can be identified even before requirements gathering has taken place. This is possible where although the exact characteristics of each component may not be known, the kinds of components required may be identified for generic applications of this type.

Levels of component that can currently be bought are:

- **Application Solutions:** whole business areas (not applicable here);
- **Sub-assemblies:** discrete parts of business areas, featuring some well-defined functionality. Typically a collection of related business components;
- **Business components:** finely granulated components that contain a small but specific piece of business functionality;

- **Technical components:** finely granulated components that contain technical functionality used to form the application architecture.

In terms of granularity, the sub-assemblies of business components will have the greatest potential for re-use, but are often not cohesive and may be difficult to assemble with others to build solutions to business problems. The smaller individual business components are likely to be more cohesive and hence more suitable for use within a specific DSDM project. These may however need to be coupled with many others to achieve significant reuse across a wider range of projects.

Such sub-assemblies and business components will have been created as a result of a business requirements analysis carried out at the *generic* level. This tends to lead to specifications and design models that focus on business rules and data, leaving the user interface to be defined as part of the assembly process. This makes DSDM ideal for the integration of these components in order to meet a *specific* business requirement, due to the nature of prototyping and user involvement inherent in the framework.

If this approach is to be successfully used within DSDM it will need to be possible to identify and acquire appropriate components from multiple sources. As this begins to happen, new sources of supply will become available and new buying criteria will evolve.

“Warehouses”, “Libraries” and “Plazas” of such components need facilities for searching for suitable components and mechanisms for storing and registering new components are becoming more sophisticated. Such repository technology is a key enabler of the component market place. For components to be usable in a DSDM project they should be described in terms of their business meaning and behaviour rather than their implementation detail. However they must be consistent with the agreed technical architecture.

If DSDM developers are to incorporate these pre-built components into their own solutions, they will need to be aware of the key business components they are likely to need in support of the processes and workflows within the scope of the business area under study. DSDM developers will work in an environment where repositories of components are browsed against a requirement to enable the assembly of many, if not all parts of any given application.

Such an approach is consistent with the way DSDM developers already work and DSDM practitioners will not need to master a different development discipline. However, it is worth considering that the need to integrate separate components together into a workable application has the potential to become complex and resource intensive.

7.2 Benefits of the assembly approach

- Time to market: early delivery of business benefit.
- Avoidance of over-engineering “analysis paralysis”.
- Team sizes and delivery scope well suited to DSDM.

It is important to note that as the revolutionary and evolutionary approaches begin to reap benefits in terms of architecture and availability of components, the assembly approach

will become the standard scenario for subsequent systems development using components.

7.3 Issues

There are a number of inherent issues in this approach, which need to be considered. These include:

- Immaturity of search engines to assist in the identification of usable and appropriate components.
- Trust in the components themselves. Testing throughout the lifecycle is built into DSDM, and will typically overcome this. However, it is important to ensure that procedures are in place to allow components to be bought on approval and ‘tried out’ within the application before a commitment to purchase them is finalised. This may involve an element of trial and error within the prototyping phases in DSDM, consistent with the ‘all changes are reversible’ principle, i.e. try before you buy. As the development progresses through a number of increments it may be possible to replace installed components by later releases with more comprehensive interfaces or with equivalent components from alternative sources. Any components, which are found to be unsuitable should be returned, any which are used should be registered as such.
- While time spent on coding within the application will be reduced by assembling from components, there will need to be a greater emphasis on testing, particularly as components are connected with each other and other parts of the development.
- Time must be allowed within the Functional Model and Design and Build Iterations to finalise any usability and ‘look and feel’ issues to ensure a consistent user interface across all components used in the development
- Component specialists may need to be drafted into the DSDM assembly team as required to assist with the identification and use of components within the project. These specialists will need to liaise closely with some sort of technical architect or co-ordinator who will have been involved with the verification of suitable components and the definition of the connections between these.
- There is unlikely to be time within a DSDM project to go through a full authorisation and certification process with purchased components. This may have to follow their use within a specific DSDM project if they are to be used regularly on subsequent projects.

7.4 Steps in the Assembly Approach

Figure 7 shows the assembly approach using the DSDM lifecycle to use components.

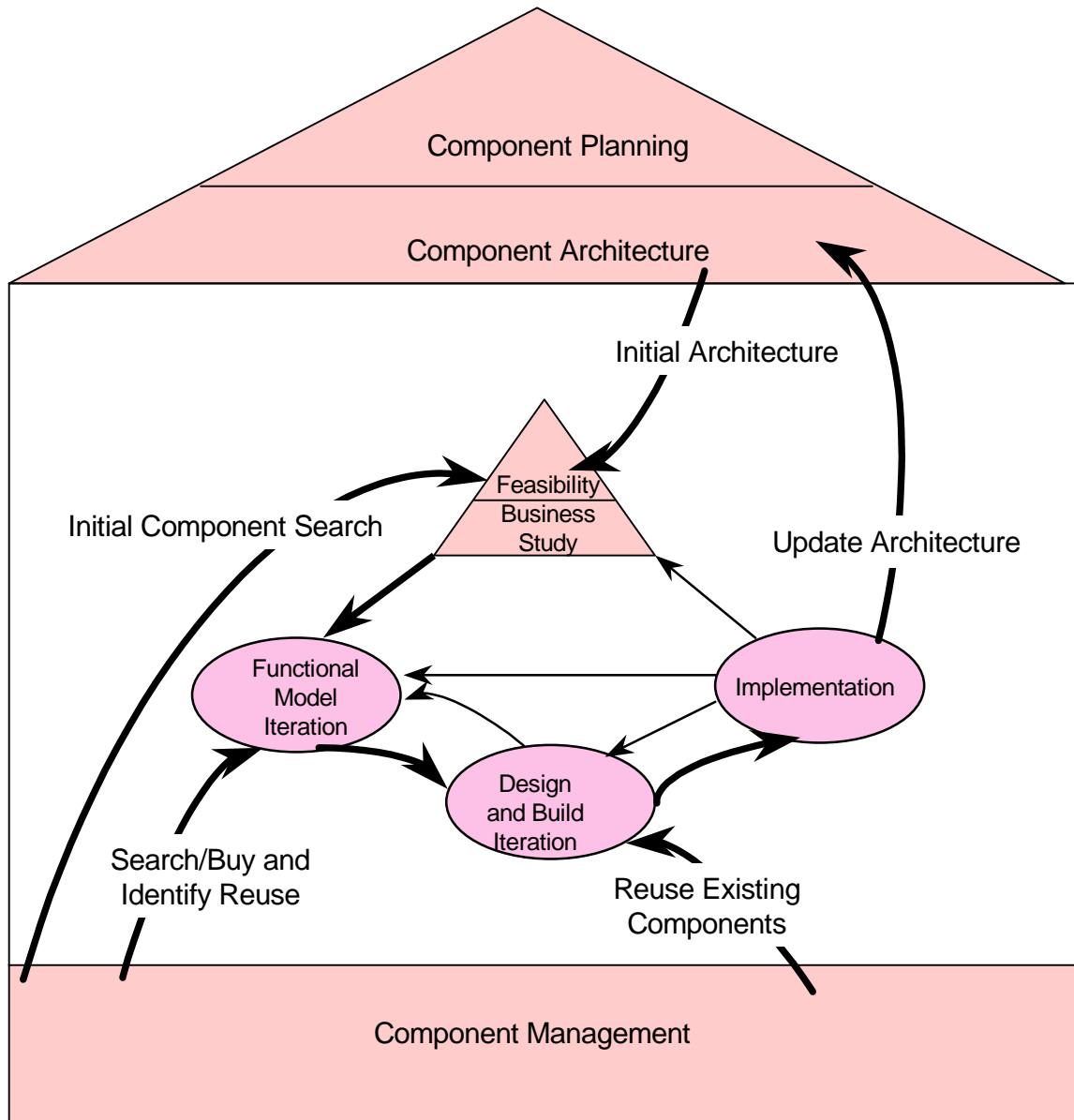


Figure 7: Assembly Process Overview

The assembly approach should perform the following steps.

- Feasibility Study
 - Undertake initial component search for generic components suitable for the business area under study. The Component Manager should be actively involved in this process.
- Business Study
 - Carry out domain modelling of the business area in conjunction with the Business Process Co-ordinator in order to identify the overall structure of the proposed system.

- **Functional Model Iteration**
 - Understand the user's description of the tasks needing support and undertake an initial catalogue search using an index to identify appropriate sub-assemblies and business components that can be integrated together to match the user's tasks and meet the overall business requirement.
 - Obtain appropriate components in conjunction with the Component Assessor and the Component Librarian ensuring both business and technical consistency with and components to be used. Assemble these components into an initial functional prototype for subsequent discussion with advisor users.
- **FMI/DBI**
 - Iteratively modify the user interface to the assembled components as a result of the prototyping by making extensions to the base components as required. If trade-offs are being made among components it is recommended that these trade-off decisions are recorded and subsequently evaluated in the final product.
- **Design and Build Iteration**
 - Integrate the agreed prototypes into a full system, undertaking synchronisation and co-ordination as necessary.

7.5 *DSDM Suitability Risk List*

Table contains a modified DSDM Suitability Risk List for use when assessing the risks of assembly CBD in a DSDM project.

Suitability Factor	Y/N	Comment
Is there commitment to user involvement?		User involvement throughout is essential in order to assess the suitability of the individual components and the integrated solution
Is the team empowered to make decisions?		A good working relationship between the development team and the Component Assessor and Librarian is vital to ensure success
Can the solution be delivered in increments?		The process of development of a solution via the assembly of components encourages iterative and incremental delivery
Will the individual teams consist of less than 6 people?		The individual teams should not exceed 6 but additional responsibilities will need to be allocated. Team will need to work closely with the architects to ensure compliance and consistency

Suitability Factor	Y/N	Comment
Can the requirements be prioritised?		Key issue of mapping available components to the requirements
Is there clear ownership?		The ownership issues in Assembly CBD relate also to IT ownership. There is a danger that the development team will not feel as comfortable with a solution assembled from components as they would with one developed by themselves
Will the project use technology suitable for prototyping?		In addition, technology needs to be suitable for identifying and managing components and their use.
Is there a highly demonstrable user interface?		The focus of the prototyping is likely to be on the user interface layer of components. Ideal for DSDM.
Will the development team have the appropriate skills		These will include the skills and knowledge necessary to assemble a solution from components. Such skills are likely to be low in early DSDM/CBD projects.

Table 3: DSDM Suitability Risk List for Assembly CBD

8 Process and Products

DSDM Phases and their associated products are shown in Table 4. Additional phases and products required for CBD are highlighted. Most of the existing products will also require enhancements for CBD.

DSDM Phase	Products
Component Planning	The Business Case for CBD Provisioning Strategy Document Quality Plan
Component Architecture	Component Architecture Model
Feasibility Study	Feasibility Report possibly supported by a Feasibility Prototype. Outline Plan
Business Study	Business Area Definition Prioritised Requirements List. System Architecture Definition. Development Plan
Functional Model Iteration	Functional Model (including a Functional Prototypes and a Non-functional Requirements List), together with supporting review and/or test records. Interface Specification
Design and Build Iteration	Design Prototypes (intermediate products), complete with supporting review records Tested System, together with supporting Test Records.

DSDM Phase	Products
	Assembly Test
Implementation	<p>User Documentation, showing how the computer system is to be used, together with identification of any supporting procedures which may be necessary.</p> <p>External and Internal Design Documentation¹</p> <p>Delivered System, together with supporting build, delivery and acceptance records.</p> <p>Trained User Population (including operators and support staff)</p> <p>Increment Review Document.</p>
Component Management	Component Inventory

Table 4: DSDM/CBD Phases and Products

¹ The external design document should include certification details that ensure quality compliance of a component for publication to the outside world.

Existing Process Checkpoints

Checkpoints for the assembly and revolution scenarios are listed in Table 5. Evolution will involve both sets of checkpoints.

DSDM Phase	Assembly	Revolution
Feasibility Study	<p>Has the problem been tackled before?</p> <p>Search for possible existing models, and frameworks. List candidates.</p>	<p>Can the problem been stated in more general terms?</p> <p>Conduct comparative studies with similar projects. Identify opportunities to generalise common features.</p>
Business Study	<p>Can generic models or frameworks be used to solve the problem?</p> <p>Extend and specialise models or frameworks.</p>	<p>Are there cost-justified opportunities to generalise common features?</p> <p>Generalise models or frameworks.</p>
Functional Model Iteration	<p>Can existing interfaces be used to solve the problem?</p> <p>Extend and specialise interfaces.</p>	<p>Can generalised interfaces be provided?</p> <p>Generalise interfaces, consider analysis patterns.</p>
Design and Build Iteration	<p>Can existing implementations be used to solve the problem?</p> <p>Reuse implementation designs.</p> <p>Purchase implementations.</p> <p>Outsource implementations.</p>	<p>Is the design flexible enough to cater for change?</p> <p>Design implementation for flexibility, consider design patterns.</p> <p>Ensure test plans cater for sufficient diversity of users.</p> <p>Can existing implementations be used to solve the</p>

DSDM Phase	Assembly	Revolution
		<p>problem?</p> <p>Reuse implementation designs.</p> <p>Purchase implementations.</p> <p>Outsource implementations.</p>
Implementation	<p>Does the implementation conform to specification of interfaces?</p> <p>Certify implementation.</p>	<p>Does the implementation conform to specification of interfaces?</p> <p>Certify implementation.</p>

Table 5: Checkpoints

9 Roles and Responsibilities

9.1 Introduction

This section contains a definition of *additional* DSDM roles for CBD projects. These do not supersede the standard DSDM roles; a CBD project will contain roles of both types.

It should be noted that although some of the standard DSDM roles do cover some of the responsibilities of the CBD roles (e.g. Technical Co-ordinator - identifying opportunities for re-use), their particular importance in CBD requires specific roles. Accordingly these responsibilities are effectively withdrawn from the standard roles for CBD developments. The standard DSDM specialist role of Component Assessor has been elevated from its specialist role to become an important CBD role - most such CBD developments will require this role.

The new roles include the following:

- **Component Sponsor**, whose job is to solicit funds and commit the organisation to CBD from an IT strategic level;
- **Component Visionary** who gains organisational enthusiasm for CBD and initiates new projects;
- **Component Programme Manager** who plans, develops and monitors CBD developments with help from the Component Architects
- **Component Architects** (Business and Technical) who design component architectures;
- **Component Assessor** who assesses re-use opportunities and selects components for assembly.

The components themselves will be implemented by the

- **Business and Technical Component Developers** who implement the components
- **Component Validator** who tests the components
- **Component Certifier** who verifies that the components meet the standards that he or she has set
- **Component Librarian** adds the certified components to the Component Library and manages them
- **Business Process Co-ordinator** works with the users to write new business processes around the new components.

A number of specialist roles are also defined; these include a **Methods Expert** who provides CBD and DSDM training and advice.

Whether a particular role is required depends on the type of CBD project being undertaken. As with the standard DSDM roles, one person may cover two or more roles, and a role may be split among two or more people.

9.2 DSDM/CBD role definitions

Component Sponsor

Skills

Similar to the standard DSDM Executive Sponsor role, the Component Sponsor must also be able to strongly influence/decide corporate IT strategies, with the ability to sell them to both the IT and Business departments.

Responsibilities

This will be typically be a high level executive technical role. The Component Sponsor is a "Component Reuse Champion" who is committed to component development and reuse. The Component Sponsor should be of sufficiently high managerial level to gain organisational commitment to CBD and to elicit funds. This is particularly important, as CBD projects will initially be of higher cost. Without this commitment, the cost effectiveness of using CBD would be compromised. The Component Sponsor must understand and be able to articulate business benefits of components in relation to costs. The role should be proactive in facilitating the right funding model. This role is likely to be combined with the Component Visionary role. Specific responsibilities are:

- Ensuring that the organisation is committed to CBD and re-use
- Soliciting funds to meet CBD costs
- Justifying the need for CBD

Component Visionary

Skills

Similar to that of the standard DSDM Visionary role, but including a good knowledge of CBD and how it is used at reference organisations.

Responsibilities

The Component Visionary helps set CBD in the business context and ensures alignment with business objectives. This is very much a reuse initiator role. The Component Visionary is responsible for arguing the case for, and introducing CBD into the organisation, influencing (and maybe becoming) the Component Sponsor role. Accordingly this team role is most appropriate when an organisation is making its first foray into CBD, and as it is introduced across the IT department. In contrast to the DSDM Visionary role, the Component Visionary does not monitor CBD projects – this is performed by the Component Manager role. Specific responsibilities are:

- Gaining organisational enthusiasm for CBD
- Initiating CBD projects
- Finding suitable CBD test-beds

Component Programme Manager

Skills

This is similar to the standard DSDM Project Manager role but also with the ability to manage multiple parallel projects. This later skill is particularly important as CBD may lead to increased outsourcing of software components.

Responsibilities

The Component Programme Manager is responsible for the planning, development and monitoring of CBD projects, and ensuring that they are run in a consistent manner. Specific responsibilities are similar to that of the Project Manager role, but at a higher level.

Business Component Architect

Skills

The Business Component Architect has similar skills to a very experienced senior business component developer, with further political awareness skills. The role must have good awareness of overall business requirements, patterns and frameworks, keep good contacts with industry bodies and have good overall vision. It also ensures consistency of design across projects and promotes the value of reuse via the reuse sponsor. It is a more strategic role than the Component Assessor is. Note: this role typically works in tandem with the Technical Component Architect.

Responsibilities

The Business Component Architect is responsible for the business architecture. The role identifies and acquires reusable business components, including the technical infrastructure software.

Technical Component Architect

Skills

The Technical Component Architect has similar skills to a very experienced senior technical component developer, with further political awareness skills. The role must have good awareness of non-functional requirements, ensuring that the technical architecture meets these requirements in a balanced way.

Responsibilities

The Technical Component Architect is responsible for the technical architecture. The role identifies and acquires reusable technical components, including the technical infrastructure software. Note: this role typically works in tandem with the Business Component Architect.

Component Assessor

Skills

The Component Assessor identifies areas for reuse improvement and pollinates reuse across solution projects. The role has good awareness of existing systems, packages, databases, and available components. This knowledge is used to assess reuse opportunities and to evaluate reuse requirements. This is a very proactive role, calling for good interpersonal skills, in approaching political situations with diplomacy and tact.

Responsibilities

The Component Assessor is responsible for monitoring the use of components in solution projects, looking for opportunities to further develop component reuse and make recommendations accordingly. The role helps solution developers to grow reuse into their solutions where appropriate as well as alert them of opportunities to reuse existing components. The role covers three areas:

- Assessing new reuse opportunities, leading to new component development
- Assessing whether software already developed, or being developed in solution projects can be made reusable
- Assessing if any existing component should be reused for a particular solution project.

Solution projects may contain a localised version of the Component Assessor role ('Component Identifier').

Business Component Developer and Senior Developer

Skills

As the standard DSDM Developer role, but including excellent business awareness of the business area in which the component covers, and an excellent working knowledge of CBD methodologies, business component development techniques and, where appropriate, of CBD tools.

Responsibilities

These roles model and interpret generic business requirements, and develop the models to produce business components. The roles are also responsible for documenting in the required detail for inclusion into the component library, and keeping the documentation up to date whenever new versions of the components are released. With the Component Assessor, these roles should ensure that the developed components have applicability, and are usable, by other component or project teams.

Technical Component Developer and Senior Developer

Skills

As the standard DSDM Developer but an excellent working knowledge of CBD methodology, business component development techniques and, where appropriate, CBD tools. Developing these components would typically require detailed low-level technical knowledge.

Responsibilities

These roles model and interpret generic technical requirements, and develop the models to produce technical service components. The roles are also responsible for documenting in the required detail for inclusion into the component library, and keeping the documentation up to date whenever new versions of the components are released. With the Component Assessor, these roles should ensure that the developed components have applicability, and are usable, by other component or project teams.

Business Process Co-ordinator

Skills

This role is expert in business process modelling and understands the relevance of the components to business needs. The role understands the effects of new, changed or acquired components across different business processes and must therefore have broad business knowledge.

Responsibilities

The role ensures tight integration between business processes and components, and demonstrates the value of components with respect to business processes. The Business Process Co-ordinator also co-ordinates the development of new or changed business processes in relation to component projects. The role is likely to be a technically aware business user.

Component Librarian

Skills

- Knowledge of component management software
- Knowledge of configuration management
- Attention to detail
- Classification skills

Responsibilities

The Component Librarian is responsible for managing the component library, ensuring that all documentation is up to date and easily accessible to developers. The Component Librarian will also be responsible for component configuration management tools.

Component Certifier

Skills

- Attention to detail
- CBD knowledge
- Quality management skills

Responsibilities

The Component Certifier is responsible for setting component specification standards, communicating them to the development teams, policing them and ensuring full compliance to these standards before any component is allowed to be stored in the Component library. The Component Certifier is also responsible for quality standards and metrics.

Component Validator

Skills

- Attention to detail

- Knowledge of testing tools and techniques
- Good business knowledge (when testing business components)
- Good technical knowledge (when testing technical components)

Responsibilities

The Component Validator is responsible for ensuring that components meet their interface and functionality specifications. Ideally, business users should test any developed business components. Additional responsibilities include developing reusable testing scripts to test new versions of any component, and developing testing prototypes.

9.3 Extra specialist DSDM CBD roles

Table 6 lists specialist roles that are additional to the specialist roles described in the DSDM Manual.

Role	Responsibility
Metrics Expert	Provides metrics guidance to projects. Collects project metrics and maintain them in a metrics database. Uses this information to provide cost benefit analysis for both projects and components.
Legacy Expert	Provides legacy application knowledge to help identify and develop legacy application wrappers and legacy integration strategies
Technical Facilitator	Facilitates between component development teams and solution teams to get agreement on component functionality and interfaces.
Methods Expert	Provides guidance on CBD and DSDM techniques. Also provides training.

Table 6: Specialist Roles

10 Roles and CBD project types

The mix of standard DSDM and specific DSDM CBD roles depends on the project type (assembly, evolution and revolution) and the maturity of the company's use of CBD i.e. a company with a mature CBD strategy will not need a Component Visionary role. The role mix is examined below.

10.1 Revolution

A revolution project will typically require most, if not all, the CBD roles specified in this section.

The standard DSDM roles required depend on the type of component being built. When building business components to meet specific business issues, an Executive Sponsor and Visionary, together with Ambassador and Advisor Users will be needed. Technical components will require a technical equivalent to these roles, but this is likely to be performed by the Component Architect and Component Assessor.

Each major component module will have an associated Project Manager/Team Leader to manage the development. Facilitator/Scribe roles are required for running workshops, documenting requirements and agreed design/interfaces.

Table 7 lists the roles used in the CBD Revolution.

Role	Used	Notes
Executive Sponsor	✓	For Business Components
Visionary	✓	For Business Components
Ambassador User	✓	For Business Components
Advisor User	✓	For Business Components
Project Manager	✓	Depending on the size of the project, may be performed by Component Manager for small projects
Technical Co-ordinator	✓	
Team Leader	✓	Depending on the size of the project, lead component developers
Developer		
Facilitator	✓	
Scribe	✓	
Component Sponsor	✓	For funding purposes

Role	Used	Notes
Component Visionary	✓	Project initiation
Component Programme Manager	✓	Project director role
Business Component Architect	✓	Business Component Design
Technical Component Architect	✓	Technical Component Design
Component Assessor	✓	Assesses component use
Business Component Developer	✓	
Technical Component Developer	✓	
Business Process Co-ordinator	✓	Co-ordinates implementation of business components
Component Librarian	✓	Manages the component library
Component Certifier	✓	Sets component standards
Component Validator	✓	Validates components

Table 7: Roles in CBD revolution

10.2 Evolution

An evolution project will need similar roles to that of assembly but with added emphasis on the additional development of components for current and future use. Consequently extra roles are required to design and develop these components. When such roles are required depends at which point in the project cycle the components are developed. In some projects component development will be considered up front whilst on others the components will be developed towards the end of the project when future applicability becomes more apparent. In the latter case, developers will re-implement business/technical specific code to make it more generic. This may be performed by either the project's core developers or a separate team of component developers.

Additional roles include the Component Sponsor to sort out additional funding issues and Component Visionary to gain the team's commitment to reuse. The additional Component Architect, Component Assessor, Component Certifier and Component Validator roles are required to design the component and ensure it meets company standards.

Table 8 shows the roles used in CBD evolution.

Role	Used	Notes
Executive Sponsor	✓	

Role	Used	Notes
Visionary	✓	
Ambassador User	✓	
Advisor User	✓	
Project Manager	✓	
Technical Co-ordinator	✓	
Team Leader	✓	
Developer	✓	Core development
Facilitator	✓	
Scribe	✓	
Component Sponsor	✓	For funding purposes
Component Visionary	✓	Project initiation
Component Programme Manager	✓	Component co-ordination management
Business Component Architect	✓	Business Component Design
Technical Component Architect	✓	Technical Component Design
Component Assessor	✓	Assesses component use
Business Component Developer	✓	Component development - may be same person as Developer above
Technical Component Developer	✓	Component development - may be same person as Developer above
Business Process Co-ordinator	✓	Co-ordinates implementation of business components
Component Librarian	✓	Manages the component library
Component Certifier	✓	Sets component standards
Component Validator	✓	Validates components

Table 8: Roles in CBD evolution

10.3 Assembly

An assembly project will typically use the standard DSDM roles with a subset of the CBD roles to support the assessment and actual use of the components as part of the solution. Table 9 shows the roles used in CBD assembly projects.

Role	Used	Notes
Executive Sponsor	✓	
Visionary	✓	
Ambassador User	✓	
Advisor User	✓	
Project Manager	✓	
Technical Co-ordinator	✓	
Team Leader	✓	
Developer	✓	
Facilitator	✓	
Scribe	✓	
Component Sponsor		
Component Visionary		
Component Programme Manager	✓	Manages component issues
Business Component Architect		
Technical Component Architect		
Component Assessor	✓	Assesses which components to use
Business Component Developer		
Technical Component Developer		
Business Process Co-ordinator	✓	Co-ordinates implementation of business components
Component Librarian	✓	Manages the component library
Component Certifier		

Role	Used	Notes
Component Validator		

Table 9: Roles in CBD assembly

11 Funding Models for CBD

11.1 CBD Funding Challenges

Return on Investment (ROI) is traditionally calculated on a project-by-project basis, in which the project owner pays for the development with common infrastructure centrally charged. Funding for CBD is considerably more challenging. For example, issues arise over ownership of shared components. Investment costs are often difficult to spread across the organisation. Diverse consumer needs need to be balanced. Political problems such as “Why should I pay for her functionality?” need to be addressed with diplomacy. The size and scope of a CBD project also dramatically affect its benefits, as does the sophistication of the organisation and whether other CBD projects have already been completed.

The underlying cause of the difficulties is that software is commonly shown as a cost (not an asset) on the corporate balance sheet. As components are intrinsically predicated on the concept of software as an asset that can be reused and extended in different contexts, this presents a major cultural challenge for most companies, the detail of which lies outside the scope of this paper. Nevertheless there are some tactical measures that we suggest here to help.

11.2 Corporate v. Project Funding

There are two major types of funding model: corporate and project. In reality a combination of the two is often sensible.

The corporate funding model assumes an enterprise-wide investment that is used to provide a jump-start to projects seeking to reduce costs through reuse of components. The advantage of this model is that because projects do not pay for component reuse, they are more likely to do it! The disadvantage is that it may be a long time before any return on investment is realised.

The project funding model assumes a basket of money provided by a group of projects with like-minded interest to reduce costs through common components. The advantage is that ROI can often be demonstrated reasonably quickly. The disadvantage is that there are risks in overlooking wider requirements and that project owners may still balk and argue about their own interests. If one owner is paying most of the money then he or she may influence requirements to the detriment of the common interest.

11.3 Funding the CBD Scenarios

The different CBD scenarios imply correspondingly different funding models. In all cases it is important to also articulate benefits, against the costs. This must include business benefits, not only software benefits.

The revolutionary approach implies a corporate funding model. Where CBD is predicated on the revolutionary approach, with the promise of long-term reusability, this necessitates more up-front investments in education, planning, architecture and enterprise analysis. There are also investment in component management tools and techniques that help to

sow suitable components. Metrics will not be available, so some rough-rules of thumb for costing is required (see 11.4).

The evolutionary approach implies a project funding model, though this may be supported by some corporate funding. The evolutionary approach to CBD can “start small” and build toward larger grained components. This has the advantage that metrics should be gathered and evolved as part of the programme.

Components by assembly imply a component funding model, though at much lower cost than the revolutionary approach. Components by assembly may appear to be “the cheap option”. However, to take the approach seriously needs considerable up-front investment in education, component management tools and techniques that help to harvest suitable components. Metrics will not be available, so some rough-rules of thumb for costing is required (see 11.4).

11.4 CBD costs and benefits

Effort and Cost Metrics

Though it is early days for components, metrics information for reusable software is starting to mature². The literature (Poulin, 1997; Tracz, 1996, McClure, 1997, Jacobson, 1997) suggests that as a rule of thumb:

- It costs between 150% and 250% more to develop reusable software than developing it with reuse
- It saves 80% to develop software with reuse than it does without reuse

These figures can be used to make high-level estimates of payback (n = the number of times the software is used):

- $$\text{ROI} = \frac{(n \times \text{saving}) - \text{investment}}{\text{Investment}} = \frac{(n \times 0.8) - 2}{2}$$
- n = 3

The reader should note that these are rules of thumb for reusable software, not components. Clearly the costs of components will be slightly higher due to increased quality costs in terms of factors like certification. However, they are at least a start.

Costs to develop a component include the following:

- Cost to determine reuse potential

² Many businesses are involved in producing costing for the supply of systems, either internally, where the IT department is treated as a profit centre, or to external clients. Introducing CBD demands a change in the typical manpower estimation approach to costing. It would be unfair to pass the full component development costs on to the first client (being 150% to 250% of the norm), but the return would not be achieved over subsequent projects as the estimates would typically run at 80% of the norm. In practice, a move to a component licensing model would have to be considered, which would entail a change in corporate policy.

- Cost to find, buy, engineer or wrap
- Cost to context test
- Cost to certify
- Cost to document

Business Benefits

Return on investment for most new approaches, including CBD, is usually phrased in terms of software productivity without thinking of real business benefits. Justification for the new approach is, from the very outset, one step removed from real business needs such as an increased ability to take advantage of new markets. This helps fuel the unhelpful but pervasive view of IT as a cost to be reduced and controlled rather than an asset to be nurtured and exploited. Table 10 shown below (Allen, 1999) is a useful first step toward countering this view by considering quality attributes commonly associated with components in terms of both the IT and business benefits that they might realise.

Quality Attribute	IT Benefits	Business Benefits
Reusability	IT productivity is enabled through reuse, not only of code, but also interfaces ³ . Reuse of legacy code through wrapping.	Time to market; reduced costs. Protect and capitalise investment in legacy assets
Maintainability.	Software problems are localised, ensuring Robust design.	Reduce maintenance costs releasing energy to focus on business need.
Accuracy.	Comprehension of specifications assisting efficacy of software process. Outsourcing of design and programming work to clearly agreed contracts.	“Fitness for business purpose” through clear interface specifications. Saving cost by utilising externally provided services.
Clarity	Rigorous interface specification	Corporate consistency and

³ Note that the same concept applies to reuse of frameworks supplied by third parties and to reuse of good design practice, as evidenced in the patterns movement.

Quality Attribute	IT Benefits	Business Benefits
	allows certification and authentication techniques to ensure that the correct interface is being used.	development of intellectual capital.
Replaceability.	Component can be easily replaced with another having different implementation code	Reduce costs in migration path to future technology.
Interoperability	Removal of dependence on system integrators to implement hybrid solutions.	Exploit business benefits of new technologies such as e-commerce.
Scalability	Management of scale and complexity; different components can be allocated to separate teams or projects; for example, isolation and management of quality measurement and testing.	Ability of business to scale-up smoothly to exploit new business opportunities; increasingly it is key niche areas, rather than traditional core business processes, that are adding most value.
Performance	Flexible installation on single or multiple nodes, allowing tuning to improve availability and response times.	Ability to respond to business needs efficiently at busy “crunch times” with peak transaction volumes.
Flexibility	<p>Liberation from “one-track” development, proprietary systems; enables hybrid solutions.</p> <p>Components can be dynamically assembled into applications at run-time rather than statically via a compile and link process</p>	<p>Leverage software market opportunities for business advantage.</p> <p>Response to business process change, including changes in workflow; components become software assets configured together to meet business needs.</p>
Adaptability	Software is readily extendible to meet new requirements.	Response to fundamental business change, including mergers and take-overs, virtual enterprises.

Quality Attribute	IT Benefits	Business Benefits

Table 10: Component quality attributes associated to their IT and business benefits

12 Bibliography

Allen, P., *Doing Business with Component-Based Development*, Application Development Advisor, 3(1), Sept/Oct 1999.

Allen, P. and Frost, S., *Component-Based Development for Enterprise Systems: Applying The SELECT Perspective*, Cambridge University Press - SIGS Publications, 1998

Cantor, M., *Object-Oriented Project Management with UML*, Wiley, 1998

DSDM Corporation, *DSDM and Software Packages*, DSDM White Paper, 1998

D'Souza, D., and Wills, A., *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison Wesley, 1999

Jacobson, I., Griss, M., Jonsson, P., *Software Reuse*, Addison Wesley, 1997

McClure, C., *Software Reuse Techniques*, Prentice-Hall Inc., N.J., 1997

OMG, *Unified Modeling Language Version 1.3*, OMG, Framingham, M.A., 1999.

Orfali, R., Harkey, D., Edwards, J., *The Essential Distributed Objects Survival Guide*, Wiley, 1996

Poulin, J.S., *Measuring Software Reuse*, Addison Wesley, 1997

Szyperski, C., *Component Software*, Wiley, 1998.

Tracz, W., *Confessions of Used Program Salesman*, Addison Wesley, 1995

Appendix A Component Definitions

“Component: A physical, replaceable part of a system that packages implementation and conforms to and provides the realisation of a set of interfaces. A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files” (OMG, 1999)

“Because components mean different things to different people, we will define the functions a *minimal* component must provide... It is a marketable entity... It is not a complete application... It can be used in unpredictable combinations... It has a well-specified interface... It is an interoperable object... In summary, a component is a reusable, self-contained piece of software that is independent of any application.” (Orfali, 1996)

“A component is an executable unit of code that provides physical black-box encapsulation of related services. Its services can only be accessed through a consistent, published interface that includes an interaction standard. A component must be capable of being connected to other components (through a communications interface) to form a larger group.” (Allen and Frost, 1998)

“A component is a unit of composition with contractually specified interfaces and explicit context dependencies only. Context dependencies are specified by stating the required *interfaces* and the acceptable execution platform(s). A component is subject to composition by third parties. For the purposes of independent deployment, a component needs to be a binary unit.” (Szyperski, 1998)

“Component (general): A coherent package of software artefacts that can be independently developed and delivered as a unit and that can be composed, unchanged, with other components to build something larger.

Component (in code): A coherent package of software implementation that (a) has explicit and well-specified interfaces for services it provides; (b) has explicit and well-specified interfaces for services it expects; (c) can be composed with other components, perhaps customising some of their properties, without modifying the components themselves.” (D’Souza and Wills, 1999)