Information Technology for Development 9 (2000) 123–139
IOS Press

Assessing important factors that support component-based development in developing countries

Douglas Kunda [*] and Laurence Brooks
Department of Computer Science, University of York, Heslington, York YO10 5DD, UK

Abstract. Most developing countries (DCs) have yet to fully benefit from the many advances in the information technology (IT) field because of specific problems experienced by these countries. Examples include a lack of systems infrastructure and resources to invest in IT. Component-Based Development (CBD) offers a number of benefits that the DCs can tap into, such as reducing development and maintenance costs and improving reuse across projects. CBD entail purchasing a number of off-the-shelf software components, each satisfying some part of the requirements of the system and integrating these components into the required system. This paper will discuss the findings of a survey conducted in a specific developing country (Zambia) to assess important factors that support the CBD process. The results suggests that, although CBD has great potential for DCs there are some social and technical factors that need to be addressed by organisations in DCs for this to be fully realised. For example, the results show that from the social point of view there is lack of management support for CBD and from the technical point of view that organisations were not familiar with the technology for integrating components.

1. Introduction

The diffusion of computer technology in many developing countries (DCs) is at an embryonic stage and does not impinge upon the overwhelming majority of people [7]. A range of factors, including infrastructural, financial, political and cultural aspects have acted against the effective development and exploitation of information technology. Redressing this situation will require significant resources and a willingness to tackle the long-term underlying causes of the problem rather than offering short-term solutions.

Institutions in DCs are turning to Commercial Off-The-Shelf (COTS) software components, primarily because modern information systems are becoming increasingly expensive to build and maintain. COTS software products are stand-alone systems that are produced by the developing organisations and sold on the open market to any customer who is able to buy them [25]. Examples include graphical user interface builders, email and messaging systems and database systems. With the development of PCs these products have increased and are cheaper because their development costs are spread across hundreds of different customers. Component-Based Development (CBD) is the process of assembling or integrating existing off-the-shelf software components into large software systems [5,6]. For example, browsers such as Netscape or Microsoft can use Adobe Acrobat as a plug-in to enhance their functionality to display more types of image formats, such as the portable document format (pdf).

The CBD approach changes the focus of software engineering from one of system specification and construction to one of identification, qualification, adaptation, integration and upgrade (for system evolution) of components. The CBD approach relies on the existence of an inventory of existing software

[*]Corresponding author. Tel.: +44 1904 432737; Fax: 44 1904 432767; E-mail: douglas@cs.york.ac.uk.

0268-1102/00/$8.00 © 2000 – IOS Press. All rights reserved

components, the emergence of component integration technologies such as Common Object Request Broker Architecture (CORBA) and Component Object Model (COM), and the development of organizational capabilities for CBD trade-off analysis and design [4]. Growing capabilities in each of these areas is encouraging the migration towards CBD in a broad range of domains.

Within CBD, in-house development and third-party development is combined: in-house development to integrate the software components and third-part development to provide the required COTS software components for integration. CBD can potentially be used to reduce software development and maintenance costs, as well as reducing software development time by bringing the system to market as early as possible [6,12]. CBD also improves reuse across programs and promotes a competitive component marketplace. CBD, therefore, has a higher potential to benefit DCs compared to other systems development strategies.

This paper presents the results of a study aimed at assessing important factors that support CBD process in Zambia. The survey involved administration of postal questionnaires and statistical data analysis. A number of important factors that support CBD were elicited. The identified factors and lessons learnt from this study will assist in elaboration and further development of a method for evaluation and selection of software components for CBD for developing countries.

2. Problems of developing software systems in DCs

Developed nations have used IT to help them change the way they do business so as to give them a strategic advantage in their operations (e.g., the use of ATMs in banks to improve customer service) [7]. However the investment returns of IT in DCs have fallen short of the potential. Mohan et al. [22] attributes these to (a) inappropriate choice of applications, and (b) inadequate consideration of the organisational and environmental factors in the formulation and implementation of the IT strategy. Manufacturers and vendors of IT and IS tend to focus on technical issues for solving the information problems of DC. However, the most critical problems are not technical; they deal with the management of this technology.

W'O Okot-uma [29] argues that the problems in introducing IT in DCs can be classified into three generic categories, namely: contextual, strategic and operational. Contextual problems are due to poor match of models of developed countries design and applications to the DCs' context, semantic discrepancies in the wording and understanding of phenomena as well as references to different value systems and different concepts of rationality. Strategic problems relate to local, national and regional policy initiatives, as reflected in the institutional intervention mechanisms of influence, regulation and implementation. Operational problems are faced by DCs due to technical and economic constraints and lack of skilled personnel.

Although identification of problems faced by DCs is a very complex task, the concentration should be on those problems that are unique to the DCs and which may have a significant impact on the assimilation of the IT. This paper highlights the problems of deficiency of skilled human resources, economic constraints, systems infrastructure deficiency and applications problems.

2.1. Skilled human resources deficiency

The lack of skilled human resources is certainly the principal barrier blocking the diffusion and effective exploitation of IT in DCs [2,7,30]. There is clearly a problem of quantity and quality (mismatch between needs of industry and trained personnel) of IT personnel [1]. A number of factors have been isolated as prime causes of these deficiencies such as evolution of technology, high turnover of skilled staff due to poor conditions of service, and lack of counterpart training under technical assistance.

2.2. Economic constraints

Economic constraints are another set of major obstacles restricting the application of IT in DCs. These include the non-existence of reliable background statistical information and inadequate capital to finance IT [29]. Several DCs suffer from both a lack of resources and a limited domestic market [1]. These countries import IT because they lack an indigenous IT industry. Scarcity of foreign currency also forces them to depend on donor agencies for much of their IT imports.

2.3. Systems infrastructure deficiency

Successful implementation of computer systems is dependent upon there being a systems infrastructure on which to build. For example, in the UK, the taxation system works comparatively efficiently because it has been built up over years and because individuals, companies, banks, accountants and so on, accept the procedures [2]. In DCs this is not the case. In most DCs [29] the electrical power utility has been intermittent and the inconveniences caused have not been negligible. DCs also often do not have adequate telecommunication infrastructure and use voice telecommunication lines for data communication, which have not proved useful.

2.4. Applications problems

The priority areas of application of computer systems in DCs are different from those of developed countries [2]. This has implications for the transferability of software. For example, DCs have the following problems and tasks:

– Development and exploitation of natural resources,
– Raising educational standards of the population,
– Raising the standards of health, and
– Increasing food production.

Bhatnagar [1] argues that the factor contributing to the low impact and penetration of computers in these economies is the type of use that such computers have been put to. Most DCs have used their computers for routine transaction processing tasks rather than strategic information systems. There is a model mismatch meaning that wrong choices of technology are often made or essential elements of technology transfer such as training are not implemented.

3. Component-Based Development

CBD focuses on building large software systems by integrating previously-existing software components [26]. In CBD [12], the notion of building a system by writing code has been replaced with building a system by assembling and integrating existing software components. After defining COTS software and CBD, this paper will then argue that CBD has the ability to reduce software development and maintenance costs and therefore would be beneficial to DCs.

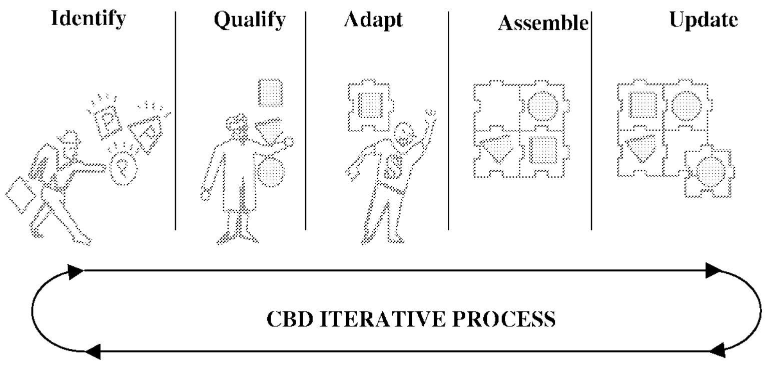

Fig. 1. Component-based development process.

3.1. *What is component-based development?*

According to Oberndorf [25] the term "COTS" is meant to refer to things that one can buy, ready-made, from some manufacturer's virtual store shelf (e.g., through a catalogue or from a price list). It carries with it a sense of getting, at a reasonable cost, something that already does the job. It replaces the scenario of developing unique system components with the promise of fast, efficient acquisition of cheap (or at least cheaper) component implementations. Examples of COTS components include Geographic Information Systems (GIS), Graphical User Interface (GUI) builders, office automation, email and messaging systems, databases and operating systems [27].

The following are some of the characteristics of the COTS software component [9,27]:

– A component software product is designed to be sold in many copies to multiple customers with minimal changes;
– The COTS software is not modified in any way but is used "as is" within the product to be developed;
– There is generally limited or no access to technical documentation, including the source code;
– There is no way to control the release of updates to the software;
– Vendor is responsible for ongoing support and maintenance – but this implies customer must accept upgrades; and
– No single customer has control over specification, schedule or evolution of the component.

Brown and Wallnau [4] define component-based systems as comprising multiple software components that:

– are ready "off-the shelf" whether from a commercial source (COTS) or re-used from another system;
– have significant aggregate functionality and complexity;
– are self-contained and possibly execute independently;
– will be used "as is" rather than modified; and
– must be integrated with other components to achieve required system functionality.

Therefore component-based development, also known as component-based software engineering (CBSE), focuses on building large software systems by integrating previously-existing software components [12]. By enhancing the flexibility and maintainability of systems, this approach can potentially be used to reduce software development costs, assemble systems rapidly, and reduce the spiralling maintenance burden associated with the support and upgrade of large systems. At the foundation of this approach is the assumption that certain parts of large software systems reappear with sufficient regularity that common parts should be written once, rather than many times, and that common systems should be assembled through reuse rather than rewritten over and over.

Examples of component-based systems can be drawn from many domains including computer aided software engineering (CASE), design engineering (CADE) and manufacturing engineering (CAME); office automation; workflow management; command and control; and telecommunications industry [4].

Building systems from COTS software components differs from other forms of re-use in that the system developer buys the components (usually without the source code) from third party developers and then integrates the components into the system [27]. In certain circumstances COTS software do not fully satisfy most of the system requirements and therefore they must be extended and tailored to satisfy local requirements. Tailoring is done by adding some element to the component to provide it with functionality not provided by the vendor [26]. Components must be adapted based on rules that ensure that conflicts among components are minimised [12]. There are a number of techniques used for modifying and extending systems, such as:

– *Application Programming Interface (API).* Most of the COTS software systems that are available in the market have some kind of an API. The developer can write a controlling program that calls the COTS component API as required [27]. As Vigder et al. [27] points out this typically involves writing a "wrapper" around the COTS component to isolate the workarounds and extensions and provide a somewhat higher level of abstraction to the component's interface. The developer then writes the main program that controls the sequence of execution including calls to the OTS component. One example of wrappers is the ODBC interface that provides an industry standard API for accessing relational databases such as Microsoft Access^{TM} [26].
– *Plug-ins.* A plug-in notifies or registers with the COTS system its capabilities and services and the system calls the plug-in as required [27]. By publishing the registration and call-back techniques, COTS vendors provide COTS users with a method of enhancing the component functionality without access to the source code [26]. Examples include browsers such as Netscape or Microsoft that use plug-ins to enhance their functionality to display more types of image formats, such as Adobe Acrobat's portable document format plug-in.
– *Scripting.* A script is an executable fragment of code, which is dynamically linked to components of the system. Examples of scripting languages include Visual Basic, AppleScript, JavaScript, tcl, Perl, and Python [27]. Early examples of scripting include simple macro languages [27]. A script can be used to extend the behaviour of a component (by having the component execute the script), or it can be used as a co-ordination mechanism to integrate two or more components (by providing the "glue" for linking the components together) [27].

Software components must be integrated through some well-defined framework. This infrastructure provides the binding that forms a system from the disparate components. There are several architectural styles and mechanisms for integrating COTS components and these include:

– *Desktop supported capabilities.* Desktops provide limited capabilities for integrating components through features such as drag-and-drop, clipboards, cut-and-paste, etc. [27]. This is generally how office automation software are integrated.
– *Data sharing – data repository.* For applications that store data in standard format, integration can be accomplished by having components read and write each other's data [27]. The idea here is that multiple components share a common data repository reading and writing the same data objects. The data repository could be flat files in the file system, a commercial database, or a repository

especially designed for component integration. A repository has four characteristics: a data storage mechanism, an interface to persistent data, an information model and a set of operations applicable to the data.

– *Messaging system – message passing.* In the message passing approach, components communicate by passing messages, informing other components of their actions, and requesting services from other components through a message broker. Currently, most active research and product development is taking place in object request brokers (ORBs) conforming to Common Object Request Broker Architecture (CORBA) standard, which provides mechanisms for language-independent interface definition and object location and activation [4].

3.2. Why employ component-based development in developing countries?

Building systems from components that are available commercially, offers an opportunity to lower costs by sharing them with other users, thus amortizing them over a larger population, while taking advantage of the investments that industry is putting into the development of new technologies [24]. Oberndorf [24] argues that most organisations typically spend far too much effort on defining to the lowest level of detail the desired characteristics of systems and how the contractors are to build those systems to achieve those characteristics. Thus a lot of resources are expended developing systems and components that often already exist or exist in "good enough" form with nearly the same capabilities – elsewhere. Therefore by employing CBD, DCs will not spend too much time by developing very expensive and inflexible systems, with only one customer to bear the development and maintenance costs over the life of the component and unable to easily capitalise on advances in new technology.

Developing CBD is becoming feasible due to the increase in the quality and variety of COTS products [12]. To date, the commercial components available which are reliable enough for operational systems, and whose interfaces are well-enough understood, have primarily been operating systems, databases, email and messaging systems, Geographic Information Systems (GIS), office automation software (e.g., calendars, word processors, spreadsheets), and Graphical User Interface (GUI) builders [12, 27]. This list is made up of applications and components that are mature and pervasive in a large number of systems both in developed countries and DCs. This maturity is the likely reason why they have been successfully marketed as COTS software components. The number of available components continues to grow, and quality and applicability continue to improve. Furthermore organisations in DCs are becoming more and more familiar with CBD as the quality and variety of COTS products increase in the marketplace.

In addition to the increasing availability of components, there are new standards that have been developed with the goal of achieving component-based development and these include the CORBA standard from OMG and COM/DCOM from Microsoft [4]. These component technologies are available on most PCs as the underlying technology. Furthermore changes in acquisition and business practices has further stimulated the move to CBD. For example, there is economic pressure on organisations to reduce system development and maintenance costs [12].

According to Voas [28], there is a belief within the industry that using COTS software components to build systems provides instant productivity gains, reduces time to market, reduces costs, fulfils organisations' mandate and has a philosophy similar to the way hardware systems are built. It is further believed that the use of standard products such as COTS software can help provide a common user "look and feel"; support system interoperability; shorter and more predictable development schedules; and reduced training and infrastructure costs. Other potential benefits of building systems from COTS include cost benefit obsolescence management and promotion of a more competitive marketplace, thereby enabling system integrators to have a wide range of choices. The desire to increase productivity and reduce costs is even stronger in DCs, therefore it is important to explore the possibility of applying CBD in DCs.

There are a number of risks associated with building systems from COTS software, including the following:

– discovering the actual technical capabilities of the COTS software components can be difficult;
– replacing older versions of COTS software with newer releases is difficult and labour intensive;
– upgrades are frequently not upward compatible; and
– the likely lack of support if the COTS provider goes out of business [9,21,26,27].

However, the emergence of component integration technology, such as CORBA and COM, makes it easier to integrate COTS software components [12]. Furthermore, DCs investing in CBD approaches, such as training of personnel will not only deal with the problem of lack of skilled human resources faced by DCs but also will enable them to fully benefit from CBD approaches.

4. Empirical study in Zambia

4.1. Goal, objectives and method

The overall goal of the study was to assess important factors that support the CBD process in Zambia. The outcome of this survey would also document and validate understanding of the current situation, as well as problems (and solutions) people have experienced in relation to CBD. Data was collected through the administration questionnaires to a sample of institutions. Questionnaires were used because they provide access to geographically dispersed samples at low cost, i.e., a large population can be surveyed relatively cheaply [10]. In addition questionnaires provide a high degree of anonymity and respondents have time to think about their answers and consult other sources.

4.1.1. Sample demographics

Zambia was chosen for the empirical study because most of the problems experienced by this country match the problems faced by other developing countries in general [7,16]. In Zambia, for example, the communication infrastructure is poor, making telephone calls expensive and beyond the reach of the common person. In addition, Zambia was chosen because of accessibility and cost of conducting the empirical study. Therefore the findings and conclusion from this study can prove useful and applicable to other developing countries.

The sources of data are personnel within institutions responsible for specifying, procuring and developing software systems. Zambian institutions were obtained by purposively sampling from the Zambia National Directory of Companies, for those that have experience in building software systems from COTS products. The questionnaires were sent to 120 potential respondents and received 14 usable responses (12%) (one was rejected because they indicated that they do not build systems from COTS software components). The job functions of the respondents were mainly management and systems analysis and they represented a wide variety of companies including financial institutions and government agencies. Most of the companies were either small or medium sized with annual turnover of not more than £100 million. The characteristics of the respondents are summarised in Table 1.

Table 1

Demographic information

Demographic information	No. of respondents respondents	Percent
Job function		
Management	5	35.7
Systems analysis	5	35.7
Application Programming	1	7.1
Operations	1	7.1
Other	2	14.3
Company primary business		
Banking/Finance	2	14.3
Insurance	1	7.1
Construction/Engineering	1	7.1
Retail/Wholesale	2	14.3
IT services	2	14.3
Government	2	14.3
Other	4	28.6
Company size		
Small (turnover less than £1 million)	3	21.4
Medium	4	28.6
Large (turnover over £100 million)	1	7.1
Don't Know	6	42.8

4.1.2. *Instrument development*

In order to ensure the validity of the measures used, several measures were taken. Many of the recommendations of Nachmias [10] were followed. To ensure content validity, an extensive survey of relevant literature was undertaken to understand the most important elements of CBD and developing countries. In content validity, the objective is to measure what is supposed to be measured, in other words to be convinced that the measurement instrument take account of essential features of CBD. To reduce the possibility of any non-random error, three faculty members of the Department of Computer Science at the University of York and two practitioners with extensive experience from UK and Zambia reviewed the questionnaire. The review focussed on the questionnaire validity (measuring the phenomena intended), completeness (including all relevant items), and readability (making it less likely that respondents will misinterpret the questions). The pilot study questionnaire also provided opportunity for the respondents to give comments explaining their responses. As a result of the pilot study feedback, some questions were further modified to improve readability and remove superfluous questions.

4.1.3. *Data analysis*

The mean (8) were calculated to measure the central tendency of the variables because, unlike the mode or median it takes into account all the values in the distribution, making it sensitive to extreme values [10]. The standard deviation (*SD*) was used to measure the extent of dispersion (variation) from the central value. However, the relative significance index (RSI) were calculated and used as a standard to compare the relative importance of the variables because of some extreme values in the responses.

$$RSI = \frac{(\overline{X})}{SD}.$$

4.2. Results and discussion

The respondents were asked to rate their strength of agreement to some factors related to building systems from COTS software. The questionnaire consisted of scaled-response from 1 (strongly disagree) to 5 (strongly agree). Tables 2 to 6 present the results of the assessment of factors affecting CBD process. The assessment of the important factors is based on comparing the relative significance index.

4.2.1. Requirements elicitation and specification

Requirements engineering covers all of the activities involved in discovering, documenting and maintaining a set of requirements for a computer-based system [25]. Requirements are defined during the early stages of system development as a specification of what should be implemented. The use of observation or ethnographic method was considered as the most significant technique for acquiring requirements (see Table 2). In the ethnographic method, the requirements engineer spends extended periods of time observing users in their normal workplace and notes made about tasks performed by participants is then analysed [20]. Maiden and Rugg [20] criticises the use of ethnography because it requires considerable time to undertake, liable to attribution errors and poses ethical problems in reporting back information given confidentially.

The use of prototyping and demonstrations was the next most highly rated techniques for requirements acquisition and specifications. In prototyping, the stakeholder is asked to comment on a prototype physical-working model of the desired system [20]. Although previous literature [19] has stressed the importance of generating scenarios and use-cases and matching them to software component, this was not highly rated by the respondents in the study. This suggests that respondents were unfamiliar with these techniques because they are new, even in the developed countries. Furthermore, the result shows that both structured methods (entity modelling) and object-oriented approaches (object and class diagrams) were considered less significant in the requirements elicitation and specification. This suggests

Table 2
Requirements acquisition and specification

Factors investigated	Mean	Standard deviation	Relative significance index
Meeting and interview	3.09	1.38	2.25
Observation	4.09	1.04	3.92
Prototyping and demonstrations	4.18	1.33	3.15
Scenarios	2.60	1.07	2.42
Rich pictures	1.80	0.79	2.28
SSM conceptual models	2.40	1.26	1.90
Decision trees and tables	2.09	1.22	1.71
ER modelling	3.36	1.80	1.86
Normalisation	3.82	1.47	2.60
Data dictionary	3.60	1.43	2.52
Entity life cycles	3.20	1.48	2.17
Data flow diagrams	3.18	1.47	2.16
Object and class diagrams	1.50	0.71	2.12
Structure diagrams	2.40	1.35	1.78
Matrices	1.55	0.93	1.65

that in DCs, the requirements elicitation and specification is not done in adequate manner. This could be because of lack of skilled human resources and lack of time (see Table 6).

4.2.2. Identification and evaluation of COTS software components

In CBD, it is important to identify COTS software components that meet the high level customer requirements and these candidate software components are then considered for a more rigorous evaluation. The respondents indicated that inventory of COTS software components is the most important activity in the identification of software components (see Table 3). Vendor advertisements and promotions followed this. In these techniques the respondents would search through vendor publications and catalogues in order to identify available COTS products. Regarding the criteria for COTS software evaluation the software quality and price were highly rated by the respondents.

In literature a number of techniques have been advocated for evaluating software components such as feature analysis, outranking, card sorting and laddering [3,15,19]. However these were not highly rated by the respondents probably because they are complex, laborious and expensive to implement without a support tool. Customer's prior knowledge and experience was the most important technique for evaluating COTS software components. This suggests that selection of COTS software components might not be conducted in a systematic manner. Paper evaluation by studying vendor documentation was the next highly rated technique for evaluation of software components.

4.2.3. Integration of software components

Components must be integrated through some well-defined infrastructure. This infrastructure provides the binding that forms a system from the disparate components. There are several architectural styles and mechanisms for integrating COTS components. In this study the highly rated architectural styles were procedure calls through API, desktop supported capabilities and data sharing each with 28.6% of respondents (see Table 4). Examples of procedural calls include components that are packaged as a procedural library, applications with an API, and databases with an SQL interface [9,27]. Desktops provide limited capabilities for integrating components through features such as drag-and-drop, clipboards and cut-and-paste. This is generally how office automation software is integrated. For applications that store data in standard format, integration can be accomplished by having components read and write each other's data. The shared data can be stored in shared files or in a shared database.

A number of new standards are being developed, driven primarily by commercial interests, with the goal of making component based software development a reality. The standards of interest include the CORBA standard from OMG and the Component Object Model (COM) from Microsoft. Although literature regards CORBA as an important standard for COTS component integration [9,27], respondents indicated Object Linking and Embedding (OLE) as the most significant technologies for integration. OLE is based on COM technology and is from Microsoft Corporation who has a monopoly on the PC operating system. This suggests that organisations use desktop supported capability techniques for integrating components. The low rating also suggests that Zambian organisations focus on adapting COTS software components rather than integrating them.

4.2.4. Organisational factors in CBD

Business environments are becoming complex and dynamic – even turbulent – leading to the need for systems with shorter lives and greater adaptability. A short-term approach can discourage CBD since benefits of CBD may only be seen after a longer period of time. Therefore the continuous change now seen in business strategy has significant impact on CBD and this was brought out by the respondents (see Table 5). Encouraging medium to long term strategies when developing systems may bring about successful implementation of CBD.

Table 3
Identification and evaluation of COTS software components

Factors investigated	Mean	Standard deviation	Relative significance index
Evaluation criteria			
Compliant with requirements	4.36	0.67	6.47
Software qualities	4.55	0.69	6.61
Availability of documentation	4.27	1.01	4.23
Maturity of COTS products	4.20	0.79	5.32
Maturity of technology	3.90	0.88	4.45
Viability of technology	3.80	1.23	3.09
Price of the COTS software product	4.45	0.69	6.48
Stability of COTS supplier	4.20	1.03	4.07
Level of COTS supplier support available	4.45	0.93	4.77
Existing relationship with supplier	3.20	1.48	2.17
Ease of migration	4.36	0.67	6.47
Political and economic factors	2.27	1.27	1.79
Conformance to appropriate standards	4.00	1.25	3.21
Ability to be tailored	2.90	1.37	2.12
Ease of integration	2.40	1.51	1.59
Identification of software components			
Prior knowledge	3.55	1.51	2.35
Inventory of existing COTS	3.82	1.17	3.27
RFPs	2.40	1.35	1.78
Market research	3.50	1.51	2.32
Fairs and shows	2.40	1.07	2.23
Adverts and promotions	3.30	1.06	3.12
Internet (Web) search	2.67	1.22	2.18
Evaluation of software components			
Study documentaion	4.18	1.40	2.98
Attend demonstration	3.55	1.37	2.59
Extensive experimentation	2.80	1.23	2.28
Customer prior knowledge and experience	3.82	1.33	2.88
User community knowledge and experience	3.82	1.25	3.05
Cards sorting and laddering	1.50	0.71	2.12
Feature analysis	2.50	1.58	1.58
Multi-criteria decision making	1.80	1.14	1.59
Outranking	1.56	0.73	2.14
Analytic Hierarchy Process (AHP)	1.78	1.09	1.63

Organisation’s work overload, skill shortages and budgetary pressure can affect the success of software systems, because it can be difficult to get management (sponsor) support if you have limited resources. That is not to say that all organisations with adequate resources will always have successful software systems. The respondents indicated that organisational resources and support is an important factor. Management support is critical to the successful implementation of any systematic method [18]. To secure management support the technical jargon related to CBD benefits should be reduced into a

Table 4

Factors that support integration of software components

Themes	Factors investigated	Respondents (%)
Architectural styles for software component integration	Component Object Model (COM)	14.3
	Distributed Component Object Model (DCOM)	0
	Object Linking and Embedding (OLE)	28.6
	Dynamic Data Exchange (DDE)	14.3
	ActiveX	7.1
	Common Object Request Broker Architecture (CORBA)	0
	OpenDOC	0
	Open Scripting Architecture (OSA)	7.1
	RMI	0
Standards for software component integration	Procedural calls through API	28.6
	Desktop supported facilities	28.6
	Message bus	0
	Data sharing	28.6
	Object request broker	0

Table 5

Organisational factors in CBD

Factors investigated	Mean	Standard deviation	Relative significance index
Customer motivation	3.18	1.33	2.40
Customer education and training	3.82	1.60	2.38
Group communication	3.27	1.56	2.10
Organisation structure and politics	3.73	1.27	2.93
Changing business strategy	4.09	1.30	3.15
Organisational resources and support	4.09	1.30	3.15
Organisational culture	3.55	1.29	2.74
External factors	3.44	1.67	2.07

simple easy to understand business case. Educating management by using an incremental approach and demonstrating to them successful case studies would also help to elicit management support.

4.2.5. Problems or constraints

According to this survey results, the respondents indicated that the main constraints, or obstacles to developing software systems, is the lack of adequately trained human resources (see Table 6). This supports the findings already identified in literature [2,7,30]. A number of recommendations have been made to address the problem of human resource deficiency in DCs, for example, Woherem [30] suggest that governments and organisations in DCs should improve salaries and service conditions for IT personnel. Furthermore, it was interesting to note that whilst literature [8,11] indicates the importance and impact of political and external issues on the success of the information system, these scored low in this survey.

Building systems from COTS software components poses a number of risks. The respondents scored highly for product mismatches and periodic releases of COTS as the major risk of building systems from COTS components. Again this is consistent with the empirical literature that replacing older versions of COTS software components with new releases is difficult, labour intensive and prone to errors [9,27].

Table 6

Problems and risks in CBD

Factors investigated	Mean	Standard deviation	Relative significance index
Constraints			
Lack of financial resources	3.08	1.44	2.14
Lack of adequate trained human resources	4.17	1.03	4.05
Lack of time	3.09	1.22	2.53
Lack of institutional support	3.17	1.59	2.00
High development costs	2.91	1.45	2.01
Political issues	1.73	1.01	1.71
External factors	1.91	1.22	1.56
Risks			
Lack of guidelines	2.73	1.56	1.75
Technical capability	3.64	1.36	2.67
Periodic releases of COTS	4.18	1.25	3.34
Loss of schedule control	2.80	1.23	2.28
Legal implications	2.45	1.29	1.90
Product mismatches	3.10	0.88	3.54
Side effects	3.11	1.27	2.45
Additional tasks	2.40	1.35	1.78
Failure to meet requirements	3.09	0.94	3.27
Lack of provider support	3.73	1.49	2.50
Difficult to select	3.30	1.16	2.85

5. Lessons learnt from the study

Although the survey sample was not large enough to draw statistically valid generalisations, nevertheless, the exercise proved to be useful and taught us a few lessons. These lessons as well as other measures reported in depth elsewhere [16] will guide our future work.

Lesson 1: Requirements elicitation and analysis is not done in a thorough manner. Building systems from COTS software components depend on successful evaluation of software components to meet user requirements. It is therefore important to elicit and know user requirements. The results from the survey suggest that there are problems with elicitation and analysis of user requirements. This is because the traditional techniques for capturing, analysing and modelling user requirements were rated low in the survey.

Lesson 2: Evaluation and selection of COTS components to meet requirements is still problematic. The respondents brought out prior knowledge and experience as the most significant techniques for evaluation and selection of COTS software components. Furthermore, selecting COTS software to meet user requirements is a problem in DCs because international aid agencies sometimes impose products from their countries on DCs. However, in developed countries appropriate techniques such as Analytic Hierarchy Process [15] and outranking [19] have been used. It is important for DCs to adopt and adapt these appropriate techniques to achieve successful implementation of component-based development.

Lesson 3: Organisations were not familiar with technology for integrating software components. Contrary to the majority of articles about the importance of CORBA and COM technology in COTS software integration this study shows that the most significant technology used by practitioners is Microsoft's OLE and DDE. In order for organisation to realise the full potential of CBD they should invest their resources in component integration technologies such as CORBA.

Lesson 4: Non-technical issues should be considered when building systems from COTS components. Organisational issues are important factors in CBD process. For example, presenting a strong business case for CBD will attract management support. This survey shows that the most important non-technical issue is consideration of organisation resources and management support. However there are other organisational issues that should be considered such as customer resistance and organisational politics. This is consistent with Clements [6] who argues that while the right architecture and other technical issues are critical to CBD success, there are also organisational, process and economic and marketing issues that must be addressed before CBD is viable.

Lesson 5: Organisation should invest in human resource development. The result indicated that lack of adequately trained human resources is the major constraint in successful implementation of CBD. Therefore, for organisations to benefit from CBD approach they must be willing to invest in training personnel in CBD approaches. Furthermore, organisations must improve the conditions of service for these trained personnel in order to avoid brain drain.

Lesson 6: Building systems from COTS components is about managing risk and therefore organisations should be willing to invest in CBD in order to realise benefits. This is because the use of COTS software components presents security vulnerabilities and other risks [12,21,26]. For example, COTS qualification for safety-critical systems is very difficult and expensive to implement.

6. Conclusion and future plans

The major outcome of this study was that the current industrial practices for CBD from a developing country (Zambia) were elicited. A number of factors that support CBD process were identified, classified and assessed (see Tables 2–6). The results suggest that, although CBD has great potential for DCs application, there are some social and technical factors that need to be addressed by organisations in DCs for this to be fully realised. For example, from the social point of view the results show that there is lack of management support for CBD and from the technical point view that organisations were not familiar with the technology for integrating components.

In order to promote and develop CBD in DCs there is a need to strengthen and improve support for the requirements acquisition, COTS software identification, qualification, adaptation, integration or assembly and upgrade (for system evolution). This can be achieved by (1) eliciting CBD best practices from developed countries and adapting them to DCs; and (2) transfer of experiences among DCs. There would also be a need to develop tools to support the use of these "best practices". For example, the systematic approach regarding COTS component evaluation and selection would prove invaluable to developing countries.

The methods and techniques from developed country are most often inappropriate to DCs because they generally do not take account of socio-cultural contexts of DCs. For example, the techniques advocated in literature [3,15,19] are too complex and laborious for DCs to adopt them. It would be more helpful to develop effective but simple, quick and inexpensive techniques for evaluating COTS components applicable to developing countries. In addition, techniques supporting CBD in DCs should achieve the following measures of success [17]:

– *High levels of system use*, as measured by polling users, employing questionnaires or monitoring parameters such as the volume of on-line transactions.
– *User satisfaction*, as measured by questionnaires or interviews. This may include users' opinions on the accuracy, timeliness and relevance of information, on the quality of service and perhaps on the schedule of operations.
– *Favourable attitudes* of users, analyst and developers about the software systems.
– *Achieved objectives*, the extent to which the system meets its specified goals, as reflected by the quality of decision making resulting from use of the system.
– *Financial payoff* to the organisation, either by reducing costs or by increasing sales or profits.

Therefore the identified factors and lessons learnt from this study will assist in elaborating and further development of a method for evaluation and selection of software components for CBD. The importance of the social dimension in IT application has been recognised in both developed [8] and developing countries [22]. For example, studies in developed countries show that the major causes of software failures are social rather than technical issues [11]. Given the impact of social issues on software, the focus for future research will be on exploring the use of socio-technical approach in CBD. The method to be defined by this research will:

1. Provide guidance for organisations in DCs regarding the process of evaluating and selecting COTS software components in CBD. For example, the framework from this research could form the basis for developing a guidebook for project and maintenance managers in DCs.
2. Provide techniques, tools and support for COTS software component evaluation phase of CBD for DCs. This will contribute to reducing risks associated with CBD.
3. Provide support for a social-technical approach to COTS software component evaluation process for DCs. For example, the current evaluation and selection of COTS components is too technical and neglects the more important social and economic factors that normally determines the choice of COTS software components.

Acknowledgements

We thank Chris Kimble and Ljerka Beus-Dukie for reviewing a preliminary draft of this paper and providing valuable comments. We also thank the Environmental Council of Zambia management for providing the funding to conduct this research. In addition, we would like to thank the Management Information Systems group in the Department of Computer Science, University of York for their support.

References

[1] S.C. Bhatnagar, ed., *Social Implications of Computers in Developing Countries*, Tata McGraw-Hill Publishing Company Limited, New Delhi, 1992.
[2] J. Bogod, *The Role of Computing in Developing Countries*, British Computer Society, London, 1979.
[3] A.W. Brown and K.C. Wallnau, A framework for systematic evaluation of software technologies, *IEEE Software* (Sept.) (1996).
[4] A.W. Brown and K.C. Wallnu, Engineering of component-based systems, in: *Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems*, IEEE Computer Society Press, Los Alamitos, CA, October 1996.
[5] Carnegie Mellon University, CBS Overview, Carnegie Mellon SEI, Available WWW (online) <URL: http://www.sei.cmu.edu/cbs/practices.html>, 1998.

[6] P.C. Clements, From subroutines to subsystems: component-based software development, *American Programmer* **8**(11), Cutter Information Corp. (November 1995).
[7] P.H. Corr, Computer education in least developed countries: A case study of Zambia, in: *Technology and Developing Countries: Practical Applications, Theoretical Issues*, R. Heeks, P. Bhatt, M. Huq, C. Lewis and A. Shibli, eds, Frank Cass & Co. Ltd., London, 1995, pp. 87–100.
[8] B. Curtis, H. Krasner and N. Iscoe, A field study of the software design process for large systems, *Communication of the ACM* **31**(11) (1988), 1268–1286.
[9] J.C. Dean and M.R. Vigder, System implementation using commercial off-the-shelf software, in: *Proceedings of the 1997 Software Technology Conference (STC 97)*, Salt Lake City, Utah, May 1997.
[10] F.C. Nachmias and D. Nachmias, *Research Methods in the Social Sciences*, 5th edn, Arnold a member of the Hodder Headline Group, London, 1996.
[11] B. Friedman and P.H. Kahn, Jr., Educating computer scientists: linking the social and technical, *Communication of the ACM* **37**(1) (1994), 65–70.
[12] C.G. Haines, D. Carney and J. Foreman, Component-based software development/ COTS integration, *Software Technology Review*, Available WWW (online) <URL: http://www.sei.cmu.edu /str/descriptions/CBSE_body.html>, 1997.
[13] P.R. Hinton, *Statistics Explained: A Guide For Social Science Students*, Routledge, London, 1995.
[14] C.B. Jaktman, Influence of organisational factors on the success and quality of a product-line architecture, in: *Conference Proceedings of the 1998 Australian Software Engineering Conference*, 1998.
[15] J. Kontio, A case study in applying a systematic method for COTS selection, in: *Proceedings of the 18th ICSE*, IEEE Computer Society, 1996.
[16] D. Kunda, Identifying current practices for COTS software – leading to best practices for developing countries, in: *Qualifying Dissertation*, Department of Computer Science, University of York, York, 1998.
[17] K.C. Laudon and J.P. Laudon, *Essentials of Management Information Systems*, Prentice-Hall, 1995.
[18] A. Lynex and P.J. Layzell, Understanding resistance to software reuse, in: *Proceedings of 1997 Software Technology and Engineering Practice*, IEEE Computer Society, 1997.
[19] N.A. Maiden and C. Ncube, Acquiring COTS software selection requirements, *IEEE Software* (March/April) (1998), 46–56.
[20] N. Maiden and G. Rugg, ACRE: selecting methods for requirements acquisition, *Software Engineering Journal* (1996), 183–192.
[21] J.A. McDermid, The Cost of COTS, *IEEE Computer* **31**(6) (1998), 46–52.
[22] L. Mohan, S. Belardo and N. Bjorn-Andersen, A contigency approach to managing information technology in developing countries: Benefiting from lessons learned in developed nations, in: *Information Technology in Developing Countries*, S.C. Bhatnagar and N. Bjorn-Andersen, eds, Elsevier Science Publishers, North-Holland, 1990, pp. 15–22.
[23] E. Mumford, *Effective Systems Design and Requirements Analysis: The ETHICS Approach*, Macmillan Press Ltd., Hampshire, 1995.
[24] P. Oberndorf, Facilitating component-based software engineering: COTS and open systems, in: *Proceedings of the Fifth International Symposium on Assessment of Software Tools*, IEEE Computer Society, Los Alamitos, CA, June 1997.
[25] I. Sommerville, *Software Engineering*, Addison-Wesley Longman Limited, Essex, 1996.
[26] M.R. Vigder and J. Dean, Architectural approach to building systems from COTS software, in: *Proceedings of the 1997 Center for Advanced Studies Conference (CASCON 97)*, Toronto, Ontario, 10–13 November 1997.
[27] M.R. Vigder, W.M. Gentleman and J. Dean, *COTS Software Integration: State of the Art*, National Research Council, Canada, NRC Report Number 39198, 1996.
[28] J.Voas, Maintaining component-based systems, *IEEE Software* (July/August) (1998), 22–27.
[29] R. W'O Okot-uma, A perspective of contextual, operational and strategy: Problems of Infomediation in developing countries, in: *Social Implications of Computers in Developing Countries*, S.C. Bhatnagar and M. Odedra, eds, Tata McGraw-Hill Publishing Company Limited, New Delhi, 1992, pp. 10–25.
[30] E.E. Woherem, IT manpower development strategy at the organisational level, in: *Information Technology Manpower: Key Issues for Developing Countries*, S.C. Bhatnagar, ed., Tata McGraw-Hill Publishing Company Limited, New Delhi, 1992, pp. 147–157.

About the authors

Douglas Kunda is the IT Manager for Environment Council of Zambia and currently a Ph.D. Student in the Department of Computer Science at the University of York, UK. He is also a student member of UKAIS, IEEE, ACM, RESG of BCS. He holds Bachelor of Engineering degree from the University of

Zambia. He has worked as an IT consultant for more then 10 years and was responsible for the development of the “Zambia Natural Resources Data Bank” a project sponsored by the World Conservation Union (IUCN).

Dr Laurence Brooks is lecturer in Management with Information Technology, teaching management courses to undergraduates on the Information Technology, Business Management and Language (commonly known as ITBML) in the Department of Computer Science, University of York. He is a board member of the UK Academy for Information Systems (UKAIS). He is a member of the IFIP and UK Northern Interest Group (NISG) for IS and Organisations. He is also one of the Principal Investigators for the University of York node on the RENOIR project and was the publicity chairman for the IEEE Symposium on Requirements Engineering (RE ’95) held in York in March 1995, and co-chaired the UKAIS annual conference in 1999.