# Cutting-edge Application Longevity: A Practical Extensionof Agile Methodology in Modernization

EkanemBassey Asuquo[1];AsuquoUwem Ekanem[2] *and* Omolu Chukwuma[3]

[1]*Computer Science Department, Delta State Polytechnic, Ozoro, Nigeria.*
[2]*Information Technology Department, BeildbestDiamond Investment Company, Uyo, Nigeria.*
[3]*Information Technology Unit, Delta State Polytechnic, Ozoro, Nigeria*

***Abstract*** *– The enormous gains in technology and innovation have been the driving force behind the massive deployment of cutting-edge applications in the society basically to deal with issues in our ecosystem particularly in social networking, communications and businesses. However, rapid technological innovations undoubtedly shortens the lifespan of some of these applications when theybecome obsolete andcould no longer be supported by modern technologies for continuous business process enhancementsand application modifications through maintenance.Component-based modernization being one of the key toolsusually adopted to modernize obsolete products also faces the challenge of difficulty in identifying and selecting quality components from the affected Appsfor reuse to ensure products longevity, hence the introduction of models like Component-based Modernization Model (CBMM) to address the situation.However, the efficacy of most of these models have not been examined. To this end, this article presents a research report from the practical experience with CBMM using a case study application deployed in microfinance banks. The Research findings indicate that CBMM is capable of securing cutting-edge application longevity by spotting unstable components that could drive the App into extinction and leading itsmodernization process towards longevity as confirmed with the actual modernization of the case study application.*

***Keywords*** *– cutting-edge applications longevity, Reusable Components, Component-based Modernization Model (CBMM)*

## I. INTRODUCTION

Technology and innovations have been the driving force behind the massive deployment of cutting-edge applications in the society basically to deal with issues that affect our ecosystem particularly in social networking, communications and businesses. However, rapid technological innovations undoubtedly shortens the lifespan of some of these applications particularly where they become obsolete and could no longer be supported for continuous business process enhancements and application modifications through maintenance.

Being that Apps with suchissues are still valuable tothe developers and users in performing the functions for which they were designed, even thoughthey are now incompatible with modern software technologies,the need for their modernization to remove the impediments becomes inevitable[1], [2], [3], [4]. Application modernization is reported as being more advantageous over its abandonment or replacement with a completely new App except for cases wherethey can no longer be evolved [5],[6] and [7].

In Reference [2] and [8], types of modernization techniquesare given as wrapping, reengineering, component-oriented reengineering, replacement, and migration with component-orientedreengineeringwidely acclaimed as capable of addressing longevity fundamental issues like Apps obsoleteness, lack of qualified engineers with experience in the obsolete tools, incomplete documentation and poor code structure [7], [8], [9].

Component-oriented reengineering as a modernization techniquehas to do with selecting stable and reusable components from a legacy APP for reuse in reengineering and modernization. The use of reusable components in modernization is of immense benefits like quick modernization process, flexibility, scalability amongst others [10]. However, the greatest challenge for component-based modernization isthe difficulty in identifying and selecting quality components for reuse in modernization [3] and [7].

The quality of reusable components selected for reuse in modernization will no doubt affect the quality of the modernized application. The big question therefore is, how best can one identify and select quality reusable components from legacy applications for reuse in modernization?Reference [7], [9] and [11] have presented some models to guide in this direction with emphasis on what constitutes quality indicators for reusable components, while othermodels provide guides

on how to select quality components for reuse from a host of components extracted from legacy applications [9], [12]and [13]. Most of these efforts are not supported with empirical data and assessment reports to actually determine their efficacies [14], [15]; even as little attempts in this direction revolve round reuse in application development and not modernization[7], [15], and [16].

In view of the above, this research was undertaken to gain practical experience in the use of CBMM in application modernization to verify the model's efficacy using a case study application coded in this research as SmartMicroeBank Application - an App used in Micro Finance banks.

## II. REVIEW OF RELATED RESEARCH WORKS

A review of related research works was undertaken to gain more understanding in the research area particularly with achievements recorded by researchers and possible gaps to fill. In [17], XIRUP modernization methodology is presented with a case study illustrations explaining the activities, related tools and findings. In [4], the problems of legacy systems and their solutions are presented with migration recommended as the best option for dealing with such. In [10], software reuse is presented as one of the best strategies for dealing with application development complexities and modernization issues.

Reference [18], proposes an approach for legacy reverse engineering to extract information needed for legacy migration utilizing the theory of business process and workflows. In [19], Component-oriented software engineering (COMPOSE), an approach built on an extensible ADL is presented as a framework for modeling, verifying and evolving legacy systems using black box components. In [20], Component-oriented modernization (COMO) meta model is presented as a tool to support a proper componentization process of legacy modernization.

Reference [2] and [8] highlight the different modernization techniques to include component-based reengineering, replacement, wrapping, screen scrapping, migration and reengineering with component-based reengineering presented as capable of yielding high quality modernized Applications compared to other techniques

In [3] and [7], Component-based modernization Model (CBMM) is presented as a tool for identifying, ranking and selecting stable reusable components from legacy applications for reuse in legacy modernization.The ranking criteria presented in these articles indicate a ranking hierarchy comprising of Highly stable, fairly stable, stable, unstable, fairly unstable and highly unstable with assigned ranks of 1, 2, 3, 4, 5, 6 respectively in the order of highest quality to the lowest quality components.

Reference [9] also presents a technique for stable components synthesis from a legacy application for reuse in modernization. In [21] and [22], the RISE Maturity Model simply denoted as RISE is presented as an effective tool for legacy components assessment and reusability in software development projects.In the model, reusability quality indicators such as stability, adaptability, flexibility, completeness, maintainability, interface complexity and understandability are presented as attributes that influence to a greater extent the quality of modernized applications.

In reference [11], the results from a review of the different reusability assessment techniques is presented with 70% of existing approaches identified as metrics-oriented for supporting object oriented development efforts. Furthermore, the model emphasizes the need for experimental approaches to provide for effective results comparison.

Also, in [23]a component ranking model which utilizes digraph techniques is presented as a veritable tool for computing the ranks of selected components and ranking same for reuse in development projects. Reference [24] provides tips on how to measure the quality of software components earmarked for reuse in software construction projects using selected software metrics that could reduce the time and efforts required in the reusability process.

Reference [25] highlights key issues to be considered in enhancing the selection process for web-based components using search engines. Moreover, it presents some useful metrics for addressing components development issues together with a technique for ranking components based on the search results.Components selection technique that is based on costs of components calculated from quality attributes of components is presented in [7]. In this case, components with cost-effective production costs are tipped as candidates for reuse.

In [16] Reuse Capability Maturity Model (RCMM) is presented as a model that emphasizes proper planning and controlling of development projects utilizing reusable components in five levels carefully articulated to ensure a steady progress of components to maturity. In achieving this, maturity goals must be clearly stated together with the corresponding activities, tasks and responsibilities. In [26], it is observed that most crosscutting frameworks employ white-box strategies in reuse processes without adequate provision for addressing knowledge requirements for architectural details and internal nomenclature. To this effect, a model-based reuse for crosscutting frameworks is presented with capabilities for assessing reuse and maintenance efforts in software projects.

Reference [27] and [28], a technique to support the specifications and serialization of planned architecture in architecture-driven modernization (ADM) context is presented. In[15], Reuse Reference Model (RRM) is presented as comprising both technical and organizational elements needed to successfully execute software reuse projects in organizations. The level of reuse as defined in RRM, could be used to determine the productivity level as well as quality and time-to-market of the organization. In [29], [30] and [31] the importance of reusability in both development and modernization projects is emphasized.

Reference [25] highlights some of the key issues involved in improving the selection support for pragmatic reuse of components provided by test-driven search engines. It also describes other metrics that are designed to address components ranking and reusability issues.

## III. FINDINGS FROM THE REVIEW

The review of related research work indicates that most of the research efforts and contributions in components reusability are mainly efforts towards determining quality indicators for reusable components and how such could be used as a guide in selecting quality components for reuse in software projects. However, the efficacy of mostof these models, methods and approaches has not been examined. To this end, this article focuses on the assessment of the efficacy of Component-based Modernization Model (CBMM) as a tool for legacy modernization using a case study application coded as SmartMicroeBank Application in the research**.**

## IV. CUTTING-AGE APPLICATION EVOLUTIONARY STAGES

Cutting-edge application deployed for use will go through three major evolutionary stages namely maintenance, modernization or replacement as indicated in the figure 1. Series of maintenance over the years will be performed to extend its usable life as it progressing from one version to another, say versions 1, 2, 3 up to N when it finally becomes legacy and subsequent maintenance hampered by some legacy fundamental characteristics listed earlier.

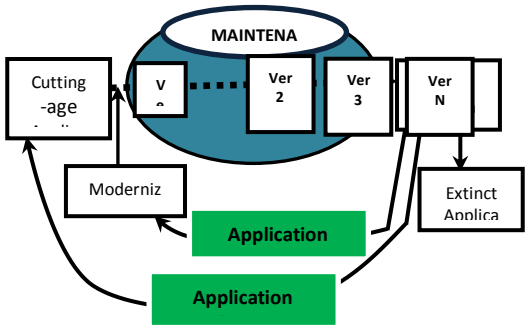


**Fig. 1. Cutting-age Application Evolutionary Stages**
**Source: [3]**

The cutting-age application that has now become legacy could be modernized to transformed it once again into cutting-edge application with maintenance impediments removed or replaced with a suitable replacement cutting-edge application; or be allowed to go into extinction where its core business values are eroded. The choice depends of the owner – the software corporation or the client.

Focusing on modernization as a viable option over replacement, withcomponent-oriented reengineering being the most beneficial modernization technique [7], stable components must be extracted from the legacy application for reuse in modernization to produce its modernized version that would continue to serve the end users. Subsequently, the modernized application will also go through series of maintenance until it once again becomes legacy and the cycle continues in that manner making the application useful with extended longevity.

Even with component-oriented reengineering as the most beneficial modernization technique, the question remains, how best should it be performed for optimum reuse of the Apps components? While this issue has been addressed by CBMM approach explained in [3] and [7], this article presents a practical application of CBMM with existing application to ascertain it's efficacy.

## V. RESEARCH METHODOLOGY

The research work was designed as case study research with the following processes:

i. Review of relevant documentations.
ii. Collection of maintenance data on each component of the case study App i.e. SmartMicroeBank Application
iii. Coding of the collected data.
iv. Computation of the Software Maturity Index (SMI) of each component for the four (4) recent versions.
v. Assessment of the stability status of each component based on their SMIs.
vi. Ranking of the components using the component assessment and ranking scheme of the CBMM
vii. Comparison of CBMM components reusability status with the reusability status of the case study components from actual modernization
viii. Results Interpretation and discussions

A review of relevant literature was conducted to establish the level of achievements in the research area as well as the research gaps. Furthermore, the maintenance data for each component in recent versions of the case study application were collected from the case study organization. These data are basically data

needed to compute the SMI of each components in the case study application. They include, number of components in the current version (M), number of added components in current version (A), number of changed components in current version (C), and number of deleted components in current version (D). Also, data on the reusability status of each component in the actual modernization were also obtained.

Using the computed SMIs of each component, the components were ranked accordingly from highly stable to highly unstable following the CBMM approach explained in [3] and [7]. The reusability status of the CBMM components were compared with the reusability status of components in the application actual modernization process. Results of the comparison were examined for facts which were presented as research findings for discussions.

## VI. DATA COLLECTION

In line with the requirements of CBMM, maintenance data of the case study App - SmartMicroeBank Applicationover a period of time with respect to four (4) recent versions of the APP were obtained from the organization and coded accordingly following the principles of the model. Table 1 presents the data collected for the purpose.

**Table 1: Maintenance Records of SmartMicroeBank Application in 4 Recent Versions**

| Component Id | Version N-3 | | | | Version N-2 | | | | Version N-1 | | | | Version N | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M | A | C | D | M | A | C | D | M | A | C | D | M | A | C | D |
| Com1 | 2 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| Com2 | 5 | 0 | 2 | 1 | 4 | 1 | 0 | 0 | 5 | 1 | 0 | 0 | 6 | 0 | 0 | 0 |
| Com3 | 2 | 1 | 0 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| Com4 | 3 | 1 | 1 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| Com5 | 2 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 0 | 0 | 0 |
| Com6 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| Com7 | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 0 | 0 |
| Com8 | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 0 | 0 |
| Com9 | 4 | 0 | 2 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| Com10 | 2 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| Com11 | 3 | 1 | 2 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 1 | 0 |
| Com12 | 3 | 1 | 1 | 0 | 4 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 5 | 0 | 1 | 0 |
| Com13 | 4 | 0 | 1 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 0 | 0 |
| Com14 | 2 | 0 | 1 | 0 | 2 | 1 | 0 | 1 | 2 | 0 | 1 | 0 | 2 | 0 | 0 | 0 |
| Com15 | 4 | 1 | 2 | 1 | 4 | 0 | 2 | 0 | 4 | 1 | 0 | 0 | 5 | 0 | 0 | 0 |
| Com16 | 5 | 0 | 1 | 0 | 5 | 0 | 1 | 0 | 5 | 0 | 1 | 0 | 5 | 0 | 1 | 0 |
| Com17 | 3 | 1 | 1 | 0 | 4 | 0 | 1 | 0 | 4 | 1 | 0 | 0 | 5 | 0 | 1 | 0 |
| Com18 | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 4 | 0 | 0 | 0 |
| Com19 | 4 | 0 | 2 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| Com20 | 4 | 0 | 2 | 0 | 4 | 0 | 1 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| Com21 | 8 | 2 | 2 | 0 | 10 | 1 | 2 | 1 | 10 | 2 | 1 | 0 | 12 | 0 | 2 | 1 |

The data collected was for four (4) recent versions of the application denoted as version N-3, Version N-2, Version N-1 and version N; where N = 4. Also, M = number of modified modules, A = number of added modules, C = number of changed modules and D = number of deleted modules. The abbreviation Com is used to denote component; being that there are 21 components in the application, they are numbered Com1, Com2 up to Com21.

The record of components reusability status of the SmartMicroeBankApplication recorded from actual modernization process is presented in table 2.

**Table 2: Components Reusability Status Recorded from Actual Modernization**

| Components Reusability Status – Actual Modernization Process | |
|---|---|
| **Components Suitable and Reused** | **Components Unsuitable and not reused but replaced** |
| Com1<br>Com2<br>Com3<br>Com4<br>Com5<br>Com6<br>Com8<br>Com9<br>Com10<br>Com12 - *reengineered*<br>Com13<br>Com14<br>Com15<br>Com16 - *reengineered*<br>Com17 - *reengineered*<br>Com18<br>Com19 - *reengineered*<br>Com20 | Com7 – *replaced* (because of required substantial changes at Front-end and back-end)<br><br>Com11 - *replaced*<br>Com21 – *replaced* |

## VII. BRIEF DESCRIPTION OF THE SMARTMICRO eBANK APPLICATION

SmartMicroeBank Application is a microfinance banking software for customers transaction processing. It is a windows menu driven application with user-friendly interfaces through which transactions are posted as the transaction occurs and the accounts updated at the ledger immediately to ensure real-time completeness of data whenever reports are queried.

The application is a client-server system, hence multiple users can have access to the same data environment at the same time and perform same type of operations say account opening from different computer

terminals or perform different types of operations from different terminals at the same time.

The initial version of the software called SmartMicroeBank version 12.2.2 was developed in 2006 and deployed for use in some Microfinance Banks. The programming language/tool used in developing the software are Microsoft Visual Basic 6.0 with Microsoft SQL Server as the database platform. Over the years many versions of the application namely versions 12.3, 12.4, 12.5, and 12.6 have been introduced inline with the required modifications to the software to meet users needs.

The newest version of the software called SmartMicroeBank version 2016.4.2is a product of the modifications made to the older version to support ISO8583 technology – international standards for electronic transaction messaging. In this case, ISO8583 interfaces were created together with corresponding modules linked to the interface. The ISO8583 interface was developed using Java programming language with the entire new version maintained with Microsoft.Net platform. Key modifications to the older version to produce SmartMicroeBank version 2016.4.2include new modules for

i. Account Balance Inquiry (e-banking)
ii. ATM Withdrawal and Related Transactions
iii. Cash Deposit and Withdrawals
iv. POS Purchase Stored Procedure
v. Bill Payment Stored Procedure.

### *A. Components of SmartMicroeBankApplication*

The case study App - has a total of 21 components coded in this research as Com1, Com2 up to Com21. A brief description of these components is given below:

**Com1 – Login Component**
This component is designed to enable users login to the eBank application using their user name and password.

**Com2 – Data Control (Buttons) Component**
This component enables users to navigate and manipulate data in any data environment accordingly. It is made up of two groups, namely data navigation buttons and data manipulation buttons. Data navigation buttons include buttons for Previous, Next, Jump, First and Last while Data manipulation buttons include Save, New, Edit, Undo, Delete and Close form buttons.

**Com3 – Record Search Component**
As the name implies, this component is designed for use to find and locate records that match specified criteria.

**Com4 – Users Setup Component**
This component is used to create a new User by specifying the user's Name, password, transaction limits, user security settings and privileges.

**Com5 – Advanced Users Setting Component**
Specifically designed for setting up users with expected user account expiration date. This group of users include for temporary staff and retirees.

**Com6 – Account Opening Component**
This is used to open a new account with customer details like name, account number, occupation and so forth entered together with customer photo and signature captured.

**Com7 – Deposit Capturing Component**
Deposit capturing components facilitates the entry of deposit transactions details into the system.

**Com8 – Withdrawal Capturing Component**
This is designed for used in entering details of withdrawal transactions made in an account.

**Com9 – Manual Journal Entry Component**
As the name implies, this component is used to pass a manual journal instruction to the accounting core module mainly for the purpose of reversal correction of wrongly posted entries example Fund Transfer Reversal Correction.

**Com10 – Sanction/Authorize Journal Entries Component**
This is used mainly to secure the approval for reversal correction made on journal entries from appropriate authorities.

**Com11 – Account Balance Enquiry Component**
This is used to check the current account balance for customers on request using account number as the parameter.

**Com12 – Journal/Voucher Cancellation Components**
This component is used to undo wrongly posted transactions. In this case, the system will cancel both the credit and debit entries in the journal.

**Com13 – Cheque Cancellation/Blocking Component**
Cheque Cancellation/Blocking Component is used to block previously issued cheque so that whenever the cheque is presented to the cashier for payment, the system will inform the cashier that the cheque has been blocked/stopped due to reasons captured during the cancellation.

**Com14 – Public Holiday Setting Component**
It is used to define public holidays declared by the government into the system. This is important since it affects the clearing days of cheques and other transactions that may have to do with public holiday or work days.

**Com15 – Credit Facility Allocation Component**
This component is used to permit credit facility like loan or over-draft on an account. This is necessary

because by default, accounts are not allowed to be over-drawn until its so permitted.

**Com16 – Budget Definition Component**
Budget definition component is used to define cost estimates (budgets) for each financial year. The budgets are linked to the various cost accounts in the system and is generated monthly, quarterly or annually with percentage variations between budget estimates and actual values indicated for assessments.

**Com17 – Financial Accounts Formatting**
This is used to define and generate any type of final accounting report ranging from P&L account, Trial Balance, Balance Sheet and CBN Return.

**Com18 – Customer Salary Scheduling Component**
This components is used for bulk posting of the salaries of customers as authorized by the organizations where they work.

**Com19 – Printing Account Listing Components**
The component is used for printing of account listing report. Account listing report is a report that displays a list of all or group of account and the date the accounts were opened on a printable report format.

**Com20 – Account Statement Component**
This is used to generate account statement for customers showing details of their debit and credit transactions over a specified period of time.

**Com21 – Application Interface Component**
This component is designed to display the application user interface through which users could gain access to the application and perform some banking transactions and related operations. It comprises of the main user interface that is linked to other forms and program modules.

## VIII. CBMM Efficacy Evaluation Using the Case Study

### Application

Component-based modernization model (CBMM) is presented in [3], [7] and [9] with the stages clearly explained using some illustrative examples. A brief summary of the stages of CBMM is given thus:

i. obtain the maintenance data of each component in the legacy application.

ii. compute the Software Maturity Index (SMI) of each component for the recent versions using the model

$$SMI = (M - (A + C + D))/M$$

Where,

M = total modules in the current version

A = total modules added in the current version

C = total modules changed in the current version

D = total modules deleted in the current version

Details on how to use the model is discussed in [3], [7] and [9].

iii. Assess the stability of each component based on its SMI using the CBMM components assessment criteria.

iv. Rank the components using the component assessment and ranking scheme of the CBMM; i.e. from Highly stable to highly unstable.

v. Select components for reuse based on their stability index

vi. Replace the unstable components with suitable components developed from the scratch or COTs.

The criteria for components assessment and ranking mentioned in steps (iii) and (iv) above are given below. A component is designated as

**Highly Stable**:if it exhibits regular SMIincreases in threerecent versions with all three SMIs tending to 1 or exactly 1. A component with thischaracteristic is of rank 1.

**Fairly Stable**: if it exhibits regular SMIincreases in threerecent versions with the last two SMIs tending to 1 or exactly 1. A component with thischaracteristic is of rank 2.

**Stable**: if it exhibitsregular SMI increases in three recentversions with the SMI of the most recent version tending to 1 or exactly 1. A component with thischaracteristic is of rank 3.

**Unstable:** if it exhibits regular/irregular SMI increases in three recent versions with the SMIs not tending to 1. A component with thischaracteristic is of rank 4.

**Fairly Unstable**: if it exhibits regular/irregular SMI decreases in three recent versions with the SMIs of the two recent versions receding from 1. A component with this characteristic is of rank 5.

**Highly Unstable:** if it exhibitsregular/irregular SMI decreases in three recentversions with theSMIs receding from 1. A component with this characteristic is of rank 6.

For clarity purposes, 0.9 is taken as a benchmark for SMI tending to 1.

### *A) Obtaining the Maintenance Data of each Component in the Legacy Application*

In applying the first stage of the model to the case study application, the maintenance data were collected from a microfinance organization and documented as indicated in table 1. Also data on reusability status of each components of the case study application in actual modernization process were collected and documented as presented in table 2.

### *B) Computation of SMIs of Components in the SmartMicroeBank Application*

Table 3 presents the SMIs of the 21 components in the microfinance bank legacy application, this were obtained using the SMI model presented above:

**Table 3: SMIs of the Application Components In the 4 recent versions**

| Component Id | Software Maturity Index (SMI) | | | |
|---|---|---|---|---|
| | Ver. N-3 | Ver. N-2 | Ver. N-1 | Ver. N |
| Com1 | 0.50 | 1.00 | 1.00 | 1.00 |
| Com2 | 0.40 | 0.75 | 0.80 | 1.00 |
| Com3 | 0.50 | 0.67 | 1.00 | 1.00 |
| Com4 | 0.33 | 0.75 | 1.00 | 1.00 |
| Com5 | 0.50 | 0.50 | 0.50 | 1.00 |
| Com6 | 0.00 | 0.50 | 1.00 | 1.00 |
| Com7 | 0.67 | 0.67 | 0.67 | 1.00 |
| Com8 | 0.67 | 0.67 | 0.67 | 1.00 |
| Com9 | 0.50 | 0.75 | 1.00 | 1.00 |
| Com10 | 0.50 | 0.50 | 1.00 | 1.00 |
| Com11 | 0.00 | 0.75 | 0.75 | 0.75 |
| Com12 | 0.33 | 0.75 | 0.80 | 0.80 |
| Com13 | 0.75 | 0.75 | 0.75 | 1.00 |
| Com14 | 0.50 | 0.00 | 0.50 | 1.00 |
| Com15 | 0.00 | 0.50 | 0.75 | 1.00 |
| Com16 | 0.80 | 0.80 | 0.80 | 0.80 |
| Com17 | 0.33 | 0.75 | 0.75 | 0.80 |
| Com18 | 0.67 | 0.67 | 1.00 | 1.00 |
| Com19 | 0.50 | 0.50 | 0.67 | 0.67 |
| Com20 | 0.50 | 0.75 | 0.67 | 1.00 |
| Com21 | 0.50 | 0.60 | 0.70 | 0.75 |

### *C) Components Stability Assessment and Ranking*

Table 4 shows the components status and ranks obtained from their SMIs assessment.Guide on how to asses and rank legacy components using CBMM is discussed in [3], [7] and [9]. For instance, the SMIs of Com1 in the 4 recent versions (see table 3) are 0.50, 1.00, 1.00 and 1.00 implying that Com1 exhibits regular SMIincreases inthree recent versions with all three SMIs tending to 1 or exactly 1. Following this assessment, it can be said to be highly stable and of rank 1.

Following the above example in applying the assessment and ranking scheme to the other components results in components status and ranks shown in table 4.

**Table 4: Ranking of the Legacy Components**

| Component Id | Software Maturity Index (SMI) | | | | Component Status | Rank |
|---|---|---|---|---|---|---|
| | Ver. N-3 | Ver. N-2 | Ver. N-1 | Ver. N | | |
| Com1 | 0.50 | 1.00 | 1.00 | 1.00 | Highly Stable | 1 |
| Com2 | 0.40 | 0.75 | 0.80 | 1.00 | Stable | 3 |
| Com3 | 0.50 | 0.67 | 1.00 | 1.00 | Fairly Stable | 2 |
| Com4 | 0.33 | 0.75 | 1.00 | 1.00 | Fairly Stable | 2 |
| Com5 | 0.50 | 0.50 | 0.50 | 1.00 | Stable | 3 |
| Com6 | 0.00 | 0.50 | 1.00 | 1.00 | Fairly Stable | 2 |
| Com7 | 0.67 | 0.67 | 0.67 | 1.00 | Stable | 3 |
| Com8 | 0.67 | 0.67 | 0.67 | 1.00 | Stable | 3 |
| Com9 | 0.50 | 0.75 | 1.00 | 1.00 | Fairly Stable | 2 |
| Com10 | 0.50 | 0.50 | 1.00 | 1.00 | Fairly Stable | 2 |
| Com11 | 0.00 | 0.75 | 0.75 | 0.75 | Unstable | 4 |
| Com12 | 0.33 | 0.75 | 0.80 | 0.80 | Unstable | 4 |
| Com13 | 0.75 | 0.75 | 0.75 | 1.00 | Stable | 3 |
| Com14 | 0.50 | 0.00 | 0.50 | 1.00 | Stable | 3 |
| Com15 | 0.00 | 0.50 | 0.75 | 1.00 | Stable | 3 |
| Com16 | 0.80 | 0.80 | 0.80 | 0.80 | Unstable | 4 |
| Com17 | 0.33 | 0.75 | 0.75 | 0.80 | Unstable | 4 |
| Com18 | 0.67 | 0.67 | 1.00 | 1.00 | Fairly Stable | 2 |
| Com19 | 0.50 | 0.50 | 0.67 | 0.67 | Unstable | 4 |
| Com20 | 0.50 | 0.75 | 0.67 | 1.00 | Stable | 3 |
| Com21 | 0.50 | 0.60 | 0.70 | 0.75 | Unstable | 4 |

As indicated in table 4 is a mix of components with various status where Com1 is highly stable (i.e. rank 1), while Com3, Com4, Com6, Com9, Com10, Com18 are fairly stable (i.e. rank 2) whereas, Com2, Com5, Com7, Com8, Com13, Com14,Com15, and Com20 are just stable (i.e. rank 3) components. Conversely, Com11, Com12, Com16, Com17, Com19 and Com21 are of unstable status (i.e. rank 4) components.

### *D) Architecture Recovery*

This was not possible as the legacy code nor was its designmade available by the software organization. Since the design was not provided, architecture recovery was based on the review of available documents with components information and their interactions with one another.

### *E) Components Classification*

Following the components rankings given earlier which is based on the SMIs, the components were further classified as stable and reusable or unstable and non-reusable as indicated in table 5. The classification is based on the fact that any component in the unstable category, that is rank 4 to 6 are unsuitable for reuse.

From the table, components in the stable categories (i.e. highly stable, fairly stable and stable) though with variable levels of stability are suitable candidates for reuse and should be treated as such. Conversely, the other six components being of unstable

status (i.e. ranks 4 to 6) should be excluded from the list of components to be reused in the modernization process.

**Table 5: Classification/Ranking of Legacy Components**

| Stable and Reusable Components | | | Unstable and Non-reusable Components | | |
|---|---|---|---|---|---|
| **Highly Stable (Rank 1)** | **Fairly Stable (Rank 2)** | **Stable (Rank 3)** | **Unstable (Rank 4)** | **Fairly Unstable (Rank 5)** | **Highly Unstable (Rank 6)** |
| Com1 | Com3<br>Com4<br>Com6<br>Com9<br>Com10<br>Com18 | Com2<br>Com5<br>Com7<br>Com8<br>Com13<br>Com14<br>Com15<br>Com20 | Com11<br>Com12<br>Com16<br>Com17<br>Com19<br>Com21 | Nil | Nil |

In other to provide the functions of unstable components not selected for the application modernized process, suitable replacement components could be redeveloped anew and reintegrated in their stead or be replaced with suitable COTS.

## IX. RESULTS AND DISCUSSIONS

The research findings as presented in table 5 indicate that fifteen (15) components from a total of twenty one (21) were stable and suitable for reuse while the remaining six (6) components namely Com11, Com12, Com16, Com17, Com19 and Com21 were not. Consequently, following the CBMM principles, suitable replacement components could be redeveloped anew and reintegrated in their stead or be replaced with suitable COTS.

Comparatively, the actual modernization process that transformed SmartMicroeBank Application into SmartMicroeBank version 2016.4.2**,** reusedall legacy components except three which were found unstable and unsuitable for reuse. The components are

i. Com7 – Deposit Capturing Component,
ii. Com11 – Account Balance Enquiry Component and
iii. Com21 - Application Interface Component

Findings further revealed that, the three components were replaced with corresponding component-oriented subsystems developed from the scratch due to their low stability and reusability status as explained by the informant. The other eighteen (18) components were reused in the modernized version after minor modifications were effected on them to further enhance their quality and suitable for the new version.

Additionally, the following four (4) new components were added to the modernized version of the App to meet the needs ofmodern eBanking transactions:

i. ATM Withdrawal component
ii. POS Processing component
iii. Bill Payment Component
iv. Internet Banking Portal Component

The summary of each componentstability and reusability status as observed with CBMM and the actual modernization process are presented in table 5.

**Table 5: Comparison of Modernization Results from CBMM and Actual Process**

| CBMM | | Actual Modernisation Process | |
|---|---|---|---|
| **Components Suitable for Reuse** | **Components Unsuitable for Reuse** | **Components Suitable and Reused** | **Components Unsuitable and not reused but replaced** |
| Com1<br>Com2<br>Com3<br>Com4<br>Com5<br>Com6<br>Com7<br>Com8<br>Com9<br>Com10<br>Com13<br>Com14<br>Com15<br>Com18<br>Com20 | Com11<br>Com12<br>Com16<br>Com17<br>Com19<br>Com21 | Com1<br>Com2<br>Com3<br>Com4<br>Com5<br>Com6<br>Com8<br>Com9<br>Com10<br>Com12 - *reengineered*<br>Com13<br>Com14<br>Com15<br>Com16 - *reengineered*<br>Com17 - *reengineered*<br>Com18<br>Com19 - *reengineered*<br>Com20 | Com7 – *replaced* (because of required substantial changes at Front-end and back-end)<br><br>Com11 - *replaced*<br>Com21 – *replaced* |

The summary indicates the following results:

i) Every components identified by CBMM to be stable and reusable were also confirmed to be so in actual modernization process and were reused accordingly except Com7 which was replaced in actual modernization process because of the need for substantial modifications both at the frontend and backend components of the APP which affected its quality.

ii) Components identified in CBMM process to be unstable and non-reusable, namely Com11 and Com21 were also confirmed to be so in actual modernization process and replaced accordingly.

iii) Other components identified in CBMM to be unstable, namely Com12, Com16, Com17 and Com19 were also confirmed to be unstable in actual modernization process but were reengineered to transform them into stable and reusable components instead of outright

replacement since a greater part of such components were still very useful and easier to reengineer.

Simply put, whatever that was declared unsuitable for reuse by CBMM was also confirmedto be unsuitable in actual modernization process except for four (4) components that were reengineered turning them into suitable components. Similarly, all components declared suitable for reuse were also found to be suitable in actual modernization except for one, that is Com7 that was replaced not because it was unsuitable but because some front-end and back-end changes substantially affected it. This practical experience clearly confirms CBMMas workable tool for ensuring the longevity of cutting-edge application and should be adopted by professionals.

## X. CONCLUSION

The importance of reusable components in software process particularly in application modernization cannot be over-emphasized. Utilization of reusable components in such processes amongst other things, facilitates quick and timely delivery of products. However, it is important to note that, the quality of the resulting product will depend greatly on the quality of the components used in the process.

To this effect, this research was designed to examine the efficacy of CBMM as a tool for legacy modernization using a case study legacy application. This clearly underscores the importance of using quality reusable components in software modernization.

Research findings clearly indicate that CBMM is an effective tool in legacy components assessment, ranking and selection for modernization where unstable and unsuitable components for reuse could be easily identified and replaced with redeveloped components of suitable COTs.

## XI. RECOMMENDATIONS

Based on the research findings, the following recommendations are necessary to further enhance the quality of modernized products:

a) CBMM is highly recommended as effective tool for legacy components assessment, ranking and selection for modernization projects where unstable and unsuitable components for reuse could be easily identified and replaced with redeveloped components of suitable COTs

b) CBMM should be designed and developed into a software tool to support effective legacy components assessment, ranking and selection for modernization.

c) Professional effortsshould be geared towards identifying components of some Apps that could be developed into reusable components and provided as COTs components to further extend the application horizon of component-based software engineering, particularly for modernization processes.

## REFERENCES


[1] S. Comella-Dorda, K. Wallnau, R. Seacord, and J. Robert, (2010). A Survey of Black-Box Modernization Approaches for Information Systems, Proceedings of International Conference on Software Maintenance.

[2] M. Saarelainen, J. J.Ahonen, H. Lintinen, J. Koskinen,Kankaanpaa, H.Sivula, P. Juutilainen, and T. Tilus,(2006). Software Modernization and Replacement Decision Making in Industry: A Qualitative Study[Online]Available At: www.bcs.org/upload/pdf/ewic-ea06-paper.pdf

[3] B. A. Ekanem and E. Woherem (2016). Dealing with Components Reusability Issues as Cutting-edge Applications Turn Legacy, Proceedings of the 2016 SAI Computing Conference, July 13-15, London, Uk [Online] Available at: www.ieeexplore.ieee.org

[4] H. S. Tontry, N. M. Murulidhir, and K. Chandrasekoram. (2017). Implication of Legacy Software System Modernization – A Survey in a changedScenarion, International Journal of Advanced Research in Computer Science [online] Available at: www.ijarcs.info

[5] B. A. Ekanem. (2015). Assessment of Components Stability for Modernization Using Software MaturityIndex, International Journal of Scientific Research and Enginerring Studies (IJSRES) 2(12) [Online] Available at: www.ijsres.com

[6] R. Khadka, B. V.Batlajery, A. M. Saeidi, S. Jansen, and J. Hage (2010).How Do Professionals PerceivedLegacy Systems and Software Modernization? UtrechtUniversity, Utrecht, The Netherlands. [online] Available at:www.servicifi.files.wordpress.com/2010/06/icse.pdf

[7] B.AEkanem, and E. Woherem (2016). LegacyComponentsStability Assessment and Ranking UsingSoftware Maturity Index, InternationalJournal of ComputerApplication (IJCA), USA. [Online]Available at: www.

[8] T. Cipresso (2010). Software Reverse Engineering Education. Master's Theses and Graduate Research, San Jose StateUniversity. USA,[Online]Available at:http://scholarworks.sjsu.edu/etd_theses/3734

[9] B. A. Ekanem (2016). A Systematic Approach to StableComponents Synthesis from Legacy Applications, International Journal of Engineering Research (IJOER),India. . [Online] Available at: www.

[10] M. Jha, and P. Maheshwari (2016). Reusing Code for Modernization of Legacy Systems, Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP'05), Researchgate, [Online] Available at:www.researchgate.net/publication/221258544

[11] C. Melo (2008). Reusable Component Identificationfrom Existing Object-Oriented Programs; World ofReuse, [Online] Available at: www.worldofreuse.blogspot.com/2008/01/reusable-component- identification

[12] S. K. Mishra, D. S.Kushwaha. andA. K. Misra (2009). Creating Reusable Software Components from Object-oriented Legacy System through Reverse Engineering.Journal of Object Technology, ETHZurich, 2009 [Online]Available at: www.jot.fm/issues/issue_2009_07/article3.pdf

[13] A. Alvaro, D. Luridio, V. C. Garcia, A. F. Prado, L. C. Travelin, E. S. Almeida (2013). ORION-RE : A Compoent-based Software Reengineering Environment ; IEEEComputer Society.
[14] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, and S. Kusumoto(2004). ComponentRank: Relative Significance Rank for SoftwareComponents Search. [Online] Available at: http://sel.ist.osaka-u.ac.jp/lap-db/betuzuri/archive/391.pdf
[15] A. Kaur, and K. S. Mann, (2010). Components Selection for Component-based Software Engineering. International Journal of Computer Applications, 2(1), 2010.
[16] B. A. Ekanem (2017). Enhancing Legacy Software Quality Through Component-based Modernization Model, Ph.D. Thesis, University of Port-Harcourt, Nigeria.
[17] R. Fuentes-Fernandez,J. Pavon, F. Gorijo (2012). A Model-driven Process for the Modernization of Component-based Systems, Science of Computer Programming, Elsevier 77(3).
[18] A. E. Roger,A. A. Ghislain and S. B. Joel (2011). Migration of Legacy Information System based on Business process Theory, International Journal ofComputer Application (IJCA), USA [Online] Available At:www.ijcaonline.org
[19] G. Kotonya, and J. Hutchinson (2008). A component-based process for modelling and evolving legacy systems; Software Process: Improvement and Practice, Wiley Online Library 13(2). [Online] Available at:www.onlinelibrary.wiley.com
[20] L. Bares, and M. Miraz (2011). A Component-oriented Metamodel for the modernization of Software Applications, 16th IEEE International Conference on Engineering of Complex Computer Systems. [online] Available At: www.researchgate.net
[21] M. Kessel, and C. Atkinson (2015). Ranking Software Components for Pragmatic Reuse. 2015 IEEE/ACM 6th International Workshop, 2015 [Online] Available at www.ieeexplore.ieee,org/xpl/articleDetails.jsp
[22] K. S. Jasmine, and R. Vasantha, (2010). A NewCapabilityMaturity Model for Reuse Based Software DevelopmentProcess; IACSIT International Journal of Engineeringand Technology 2(1), 2010
[23] S. Younoussi, and O. Roudies(2015). All AboutSoftware Reusability: A Systematic Literature Review; Journal of Theoretical and Applied InformationTechnology, 2015, [Online] Available at:www.jatit.org
[24] V. Garcia, D. Lucredio, and A. Alvaro (2007). TowardsA Maturity Model for Reuse Incremental Adoption,Proceedings of Simposio Brasileiro de Componentes, Arguitetura e Reutilizacao de Software (SBCARS), 2007
[25] B. A. Ekanem, E. Woherem and J. E. Amadi-Echendu (2016). On Extending the Usable Life of Legacy Software,International Association for Management of Technology (IAMOT) 2016 Conference Proceedings, Florida, USA. [Online] Available at: www.iamot2016.org
[26] T. Gottardi, R. S. Durelli, O. P. Lopez, V. V. Camargo (2013). Model-based reuse for crosscutting frameworks : Assessing Reuse and Maintenance Efforts, Journal of Software Engineering and Development, Springer Berlin Heidelberg.
[27] A. S. Landi, F. Chagas, B. M. Santos, R. S. Costa, R. Durelli, R. Terra, V. V. Camargo (2017). Supporting the specifications and Serialization of Planned Architectures in Architecture-driven modernization context. 2017 IEEe 41st Annual Computer Software and Application Conference (COMPSAC).
[28] R. Perez-Castillo, I. G. Guzman, and M. Piattini (2011).Knowledge Discovery Metamodel-ISO/IEC 19506: AStandard to modernize legacy systems; Computer Standard and Interfaces, 33(6), 2011.
[29] V. Subedha, and S. Sridhar, (2012). Design of DynamicComponent Reuse and Reusability Metrics Library forReusable Software Components in Context Level,International Journal of Computer Applications, 40(9): 30-34, 2012, [Online]Available at: www.ijcaonline.org
[30] F. Fazal-e-Amin, A. K. Mahmood, and A. Oxley (2011). A Review of Software Component ReusabilityAssessment Approaches; Research Journal ofInformationTechnology 3(1), 2011. pp. 1-11
[31] A. Malinova (2010). Approaches and Techniques forLegacy Software Modernization, Bulgaria ScientificWorks, 37(2), University of Plovdiv, Plovdiv, Bulgaria. [Online] Available at: www.fmi.uni-plovdiv.bg/GetResource?id=402