

# Visionet IT Modernization Empowering Change

A Visionet Systems White Paper  
September 2009

**Visionet Systems Inc.**  
3 Cedar Brook Dr.  
Cranbury, NJ 08512  
Tel: 609 360-0501

## Table of Contents

1	Executive Summary.....	4
2	Introduction .....	4
3	Current Practices, Problems, and Solution .....	5
3.1	Modernization Approaches .....	5
3.2	Application Conversion.....	5
3.3	Application Re-Model .....	6
4	The Visionet System’s IT Modernization Process .....	6
4.1	Key Challenges.....	7
4.2	Planning .....	7
4.3	Construction .....	9
5	Visionet System’s Code Miner – Legacy Code Comprehension Tool.....	10
5.1	Parser.....	10
5.2	Code Analyzer .....	11
5.3	Code Visualizer.....	11
5.4	CodeMiner <sup>®</sup> Output.....	11
6	Conclusion.....	13

## Table of Figures

Figure 1:	Acting forces pressing IT modernization need.....	4
Figure 2:	Legacy system comprehension cycle.....	8
Figure 3:	Architecture Assessment & Re-Modeling.....	9
Figure 4:	Legacy System Comprehension and TDD.....	9
Figure 5:	Brief Modernized System Development Process.....	10
Figure 6:	CodeMiner <sup>®</sup> functional blocks.....	10
Figure 7:	CodeMiner <sup>®</sup> Parser.....	11
Figure 8:	CodeMiner <sup>®</sup> Code Analysis.....	11
Figure 9:	CodeMiner <sup>®</sup> Visualization.....	11

*Organizations with state of the art IT infrastructure a few years ago are struggling now to keep up with the very system due to ever declining numbers of IT workers with skills to keep them running.*

## 1 Executive Summary

IT organizations want to reduce total cost of keeping up with infrastructure but increase ability to react to business changes & demands. Reliance on legacy skill set can increase maintenance costs over time and reduce agility to change, forcing IT organizations to consider modernization.

IT modernization that supports a consummate migration process will allow organizations to take advantage of modern technologies and still preserve business content and continuity of the organizational processes.

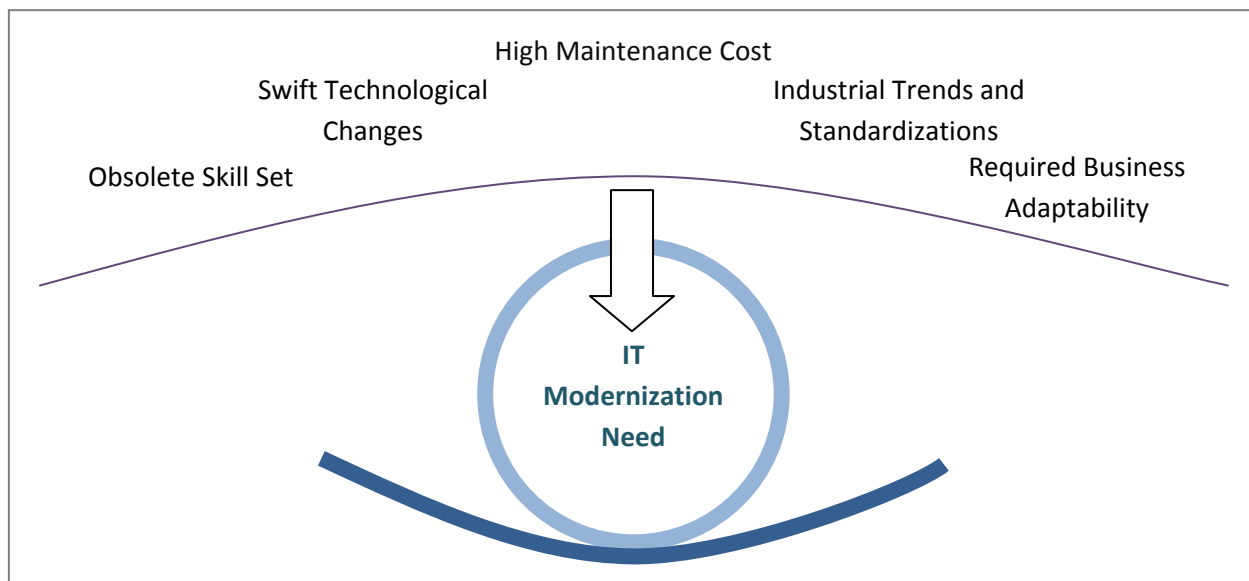


Figure 1: Acting forces pressing IT modernization need.

## 2 Introduction

Rapid influx of new technology and business needs has always been a challenge for IT shops. While newer technologies provide greater agility and enable organizations to meet new business needs, they also obsolete existing technical skills that have in-depth domain knowledge. Businesses have traditionally relied on application retirement and replacement with off-the-shelf packaged software solutions, but for many such a path is not an option because either there are no off the shelf package available or the adoption of one would entail loss of competitive advantage. The only option available

for these businesses has been to go through a re-write using resources that are technically proficient but domain poor or domain proficient but technically poor.

To meet these challenges there is a need to adopt a process, with appropriate tools and methodology that enables domain learning at a rapid pace in order to reduce the risks associated with migration.

### 3 Current Practices, Problems, and Solution

There are a number of approaches that are available for migration of legacy systems, these approaches are examined in this section and their comparative merits are evaluated to identify the best practice.

#### 3.1 Modernization Approaches

There are a number of approaches available for legacy modernization, where each approach addresses a particular situation that may apply, these approaches are:

1. **Application Retirement:** The current business need for the application is assessed and evaluated if the application is still needed or is cost beneficial to migrate, if it turns out to be the case then the application is retired and migration is avoided.
2. **Application Replacement:** In many situations a suitable off the shelf software package is available that can be used in place of the existing legacy application. The application is customized and deployed instead of migrating the legacy application.
3. **Application Conversion:** One approach to migration has been through automated conversion of the legacy system and language to a new platform and language. These situations are suitable where the running cost of the new platform is substantially lower than the legacy platform, so it is worthwhile to take the same application code and re-platform it than it is to re-architect it.
4. **Application Re-model:** In order to benefit from new technology and features offered by modern platform there is little option but to re-model an existing application for the new target architecture. In order to carry out a re-modeling project effectively, skills in both the legacy and the target platform are required in addition to domain knowledge. Which makes re-modeling the most difficult option, yet at the same time the best option for migration.

#### 3.2 Application Conversion

Application converters are typically lexical conversion tools that convert the source language to the target language with some degree of manual intervention. These tools use the support of custom libraries that bring the target language and platform closer to the source in order to ease the conversion complexity. By doing this they introduce into the target platform some of the limitations and shortcomings of the source platform.

Typically after application is converted the resulting code needs to be re-factored before a maintainable and readable version is available, this refactoring eliminates most of the gains made by using an automated tool, and yet the end result may still have some artificial limitations left over which may limit the adoption of modern features that was originally sought.

### 3.3 Application Re-Model

Application re-modeling requires the re-design of the application from scratch on the target platform based on the legacy application business functionality. Taking this approach no artificial limits are introduced, and the target platform can be fully leveraged to meet future business challenges.

The major impediment to re-modeling is the lack of knowledgeable resources in the target platform, however this can be sufficiently addressed by using resource from service providers. However, this introduces a new dilemma where the newly hired staff do not have the required domain knowledge, and the existing staff do not have the required technical knowledge.

One approach is to go back to the business users can capture the system requirements from scratch, however that in most cases is not a viable option. Visionet offers an alternative approach through it's CodeMiner<sup>®</sup> tool where business analysts can quickly "learn" the functionality of a system and leverage the tool to develop detailed use cases of the application. Armed with these use cases the application can now be designed and developed on the target platform.

## 4 The Visionet System's IT Modernization Process

Visionet Systems with extensive experience gained over many years with legacy migrations, has developed a Three Step methodology for migration of legacy systems. These are:

1. **Planning** – During this phase the system functionality is defined and modeled for the target platform. Using the functional specifications and the technical specifications of the system, the project plans and test plans for the system are prepared, and based on the details in these plans, we can offer our customers fixed price proposals for the construction of the system.
2. **Construction** – One the plans are approved and in place, the application construction takes place. The key documents used throughout the construction phase are the functional specifications and technical specifications of the system. Also included in this phase of the project is the user acceptance testing of the final delivered product.
3. **Implementation** – Once the application has been tested it is ready for implementation. The details of the implementation depends on a variety of factors, such as project phase distribution, application inter-dependencies, parallel run requirements, etc. Based on all these requirements an implementation plan is drawn up during the Planning step and that plan is put into operation during the Implementation step.

In this methodology the key step is the planning phase where the project plans, detailed technical designs and other important project decisions are made. Within the planning phase the functional specifications document is the corner stone of all the planning and estimation activities.

In standard SDLC projects the functional specifications document is derived from the user requirements document and requirement elaboration sessions. In the case of application modernization the requirements gathering phase is not deemed necessary as the existing legacy application *already* implements the required functionality. Hence it is expected that the requirements can be derived from the existing source code.

## 4.1 Key Challenges

There are many challenges involved in the planning phase of an application migration project. The key challenges are described here:

**Documentation:** Application documentation can be a good starting point for developing an understanding of a system, however documentation for most applications tend to be limited or out of date.

**Mentors:** By and large application domain knowledge is transferred by tapping into the knowledge of peers. However, in some cases these peers may not be available or knowledgeable enough about the application.

**Users:** Another commonly used method is to tap the users, especially power users, of the application to gain domain and operational knowledge of the functionality. Typically, end users are also busy with their own day to day activities and have very limited time available. In addition if the migration project requires heavy end user involvement that is perceived as a major impediment to kick off the project in the first place.

**Source Code:** Trying to understand an application from the source code is a difficult task, primarily because the source code is at a machine level abstraction which is not intuitive for human level of understanding. In addition human understanding is developed through layers going from progressively higher to lower levels of abstraction. This issue is further compounded by the fact that a person can absorb a limited amount of new information at any one time, however as the learning of a domain increases the persons capacity to absorb new information related to that domain also increases exponentially.

## 4.2 Planning

Visionet Systems with extensive experience gained over many years with legacy migrations, has developed processes and tools that enable the layered learning required by engineers working on

application source code directly without the benefit of system documentation, while at the same time reducing dependency on business users and legacy skill sets.

The CodeMiner<sup>®</sup> tool is a visualization and analysis toolkit, that facilitates understanding of a system through graphical representation of the system functions, and analysis of the source code to discover interdependencies and interactions of various system components. Information gathered by CodeMiner<sup>®</sup> is stored in a database that can easily be queried and analyzed by the business analysts at any time.

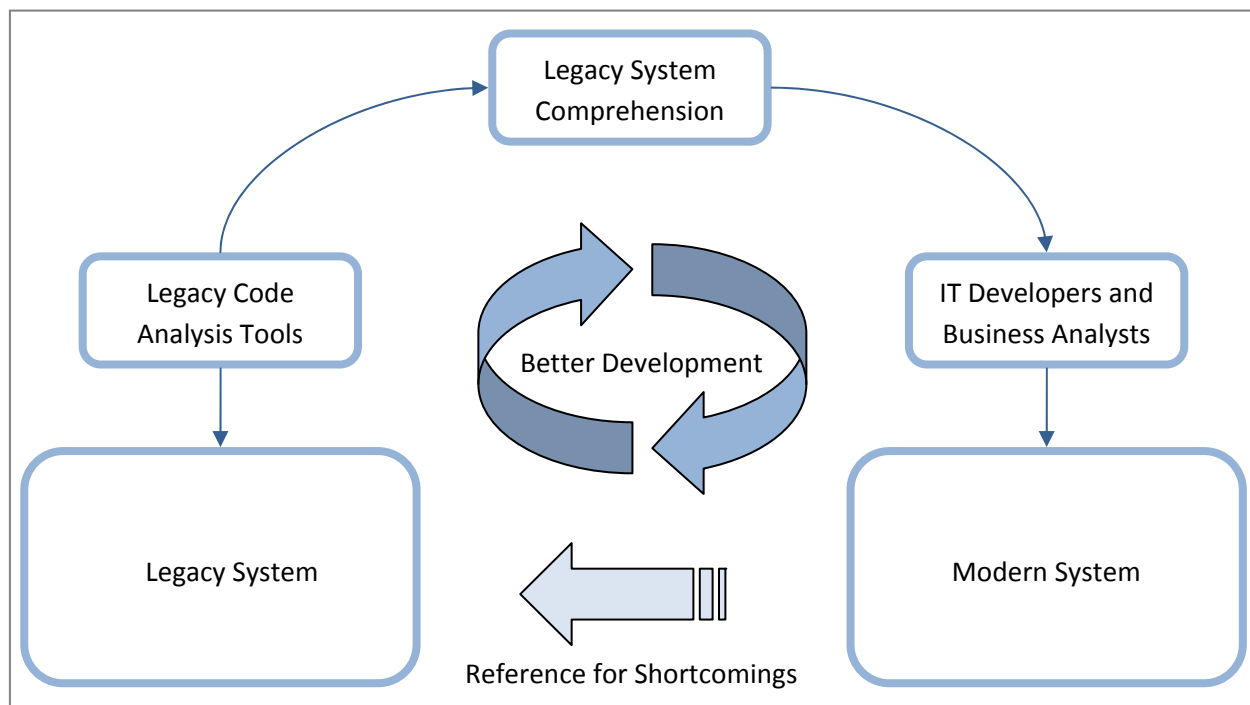


Figure 2: Legacy system comprehension cycle.

Working back from the information gathered by CodeMiner<sup>®</sup>, business analysts can re-create the higher level architecture of the system by looking at the program to program inter-dependencies and interactions. This information coupled with the application portfolio information gathered in collaboration with the IT staff will establish the architecture of the system as it stands today and help identify key areas of the applications.

The architectural assessment helps prioritize the applications in the portfolio in order to segmentize and sequence application modules for migration and deployment.

To understand the system architecture the analyst will require to carry out an in depth analysis of the application, in order to better comprehend the rationale for the architectural design choices made. CodeMiner<sup>®</sup> ability to comprehend a mixture of language and technology environments, helps the business analyst in understanding and discovery without requiring expertise in the underlying language

and technology used, which in turn affords a significant productivity boost while lowering the complexity barrier.

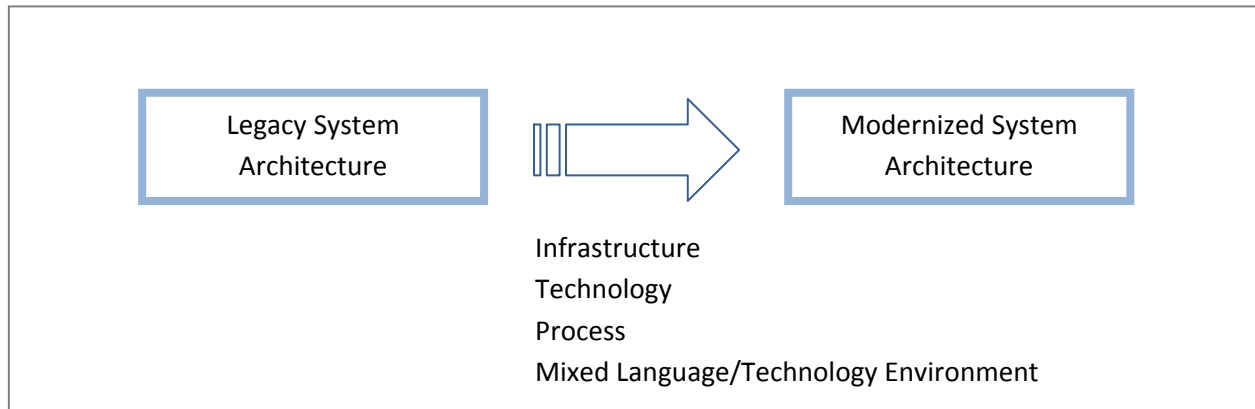


Figure 3: Architecture Assessment & Re-Modeling

### 4.3 Construction

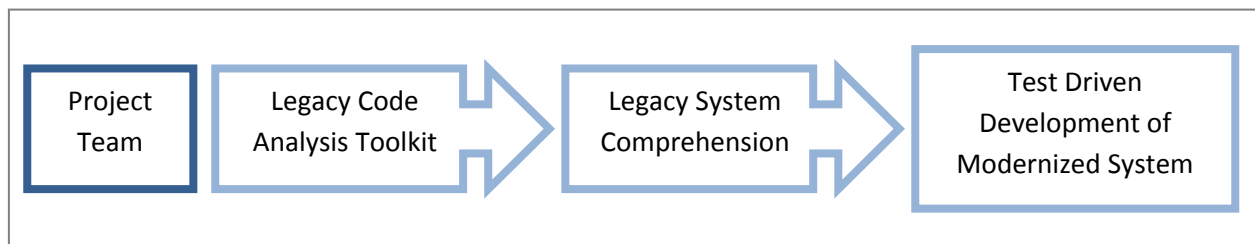


Figure 4: Legacy System Comprehension and TDD

Having completed the planning and covered the architecture of the old and completed the segmentation process the project team comprising of essential target platform experts will then proceed with the construction of the new system.

Visionet believes in a test driven methodology where the components being migrated are developed based on test cases designed for that component functionality. This will help incubate the system in a layered manner as new functionality is developed.

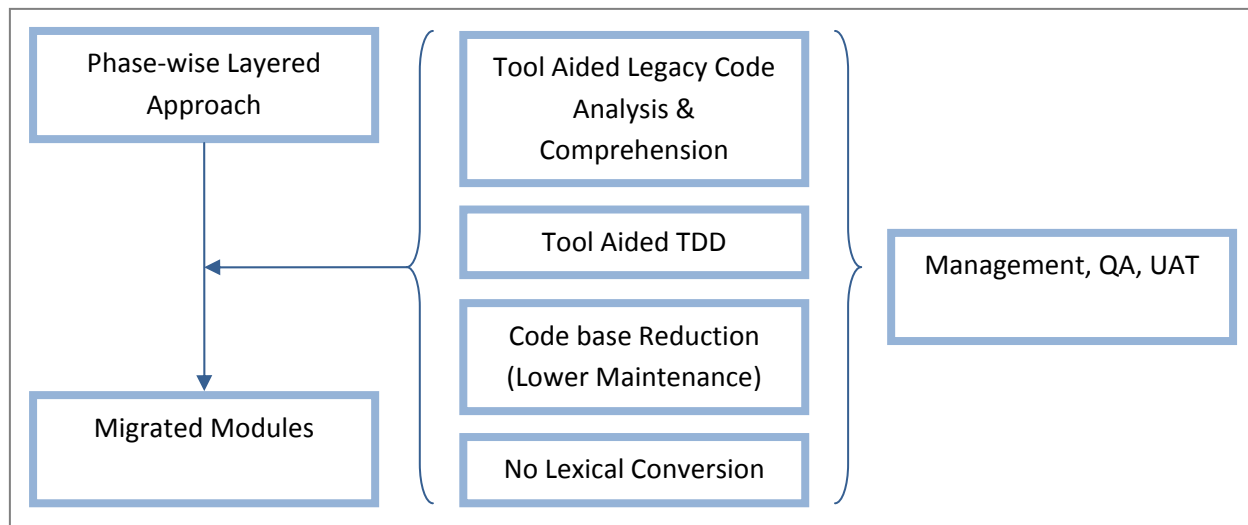


Figure 5: Brief Modernized System Development Process

## 5 Visionet System's Code Miner – Legacy Code Comprehension Tool

CodeMiner<sup>®</sup> is a visualization and analysis toolkit, used to facilitate the comprehension of a system through graphical representation of the system functions, system interactions and dependencies.

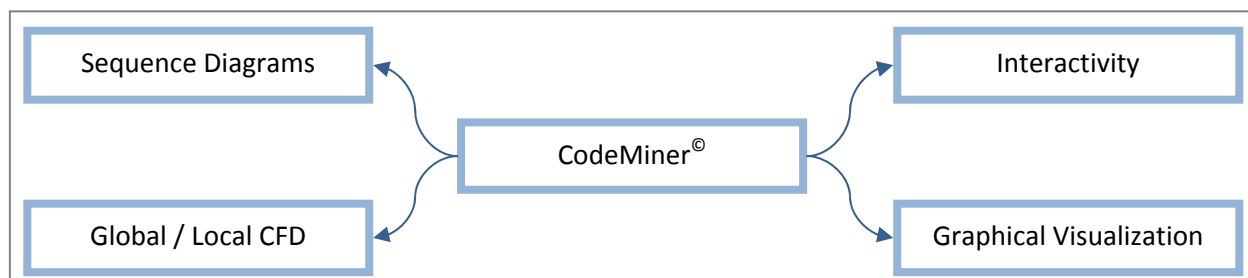


Figure 6: CodeMiner<sup>®</sup> functional blocks.

The application source code is mined for information using a parser to parse the source code and a code analyzer to extract the required information. CodeMiner<sup>®</sup> broadly has the following modules:

1. **Parser:** An LALR parser for parsing the source language and generate the abstract code information.
2. **Code Analyzer:** This is the core module of CodeMiner<sup>®</sup> where the abstract code information is processed and required information is extracted and stored in a database.
3. **Code Visualizer:** The extracted information is read from the database and comprehension graphs are generated to facilitate the understanding of the system.

### 5.1 Parser

CodeMiner<sup>®</sup> utilizes state of the art LALR algorithm for parsing the source code. The grammar for the language is described in the BNF syntax for the parser. The parser takes the grammar as input and parses the source code to generate the abstract code information used by the code analyzer.

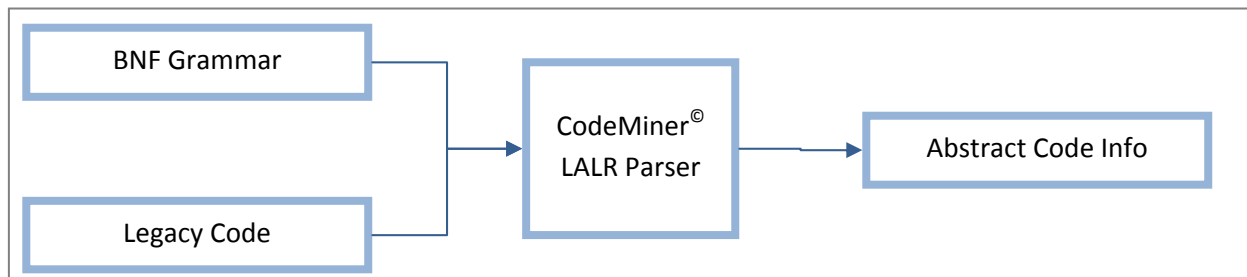


Figure 7: CodeMiner<sup>®</sup> Parser

## 5.2 Code Analyzer

The abstract code information generated by the parser is stored in a database where the code analyzer reads and processes the information to extract specific information about the source code and its interdependencies and generates various graphical representation of the information as needed by the code visualize module.

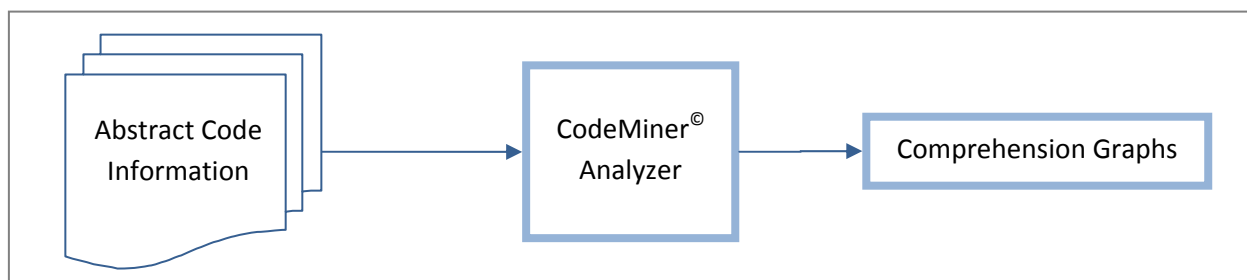


Figure 8: CodeMiner<sup>®</sup> Code Analysis

## 5.3 Code Visualizer

The code visualize is a GUI based tool that allows the user to interactively navigate the code to extract information and aid comprehension with the help of syntax highlighting and viewing different graphs to bring into focus different aspects of the code and its relationships.

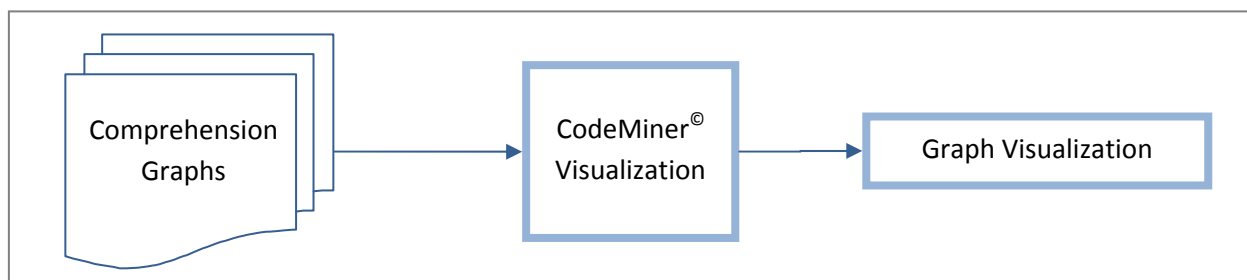


Figure 9: CodeMiner<sup>®</sup> Visualization

## 5.4 CodeMiner<sup>®</sup> Output

Aside from storing the abstract code information in a referenceable database for end user analysis, CodeMiner© also has a number of built in reports that are available. These include:

### 1. **Business Process Graphical Visualization**

CodeMiner© has the capability to identify the business functions embedded in the code through the use of various intelligent heuristic algorithms. This helps the business analyst to concentrate on the business function of the code rather than sifting through the code and architectural constructs.

### 2. **Data Structure Relationship Analyzer and Adviser**

The Data Structure relationship analyzer, identifies the data structures that are used in the code and their relationships. This feature is useful to identify the structure of flat files and non-relational program described data. This information is then used to create the underlying data structure for the files which can then be used to create an Entity Relationship model for the migrated code.

### 3. **Expert System for Filtering out database, I/O calls**

Through the use of expert systems fine tuned for the target language, CodeMiner© traces all the database and I/O calls and identify where and how database fields are being used. This helps the analyst trace through the life cycle of the file fields and identify processing and reading programs, in addition code layers are easily identified from an otherwise un-layered architecture.

### 4. **Filter based Code Structure Visualizer.**

CodeMiner© also has the ability to filter out certain code structures in order to enable the analyst to focus on areas of interest within the application. This tool is invaluable for large monolithic programs that need to be dissected.

### 5. **Function Call hierarchy Graph**

CodeMiner© call hierarchy graph shows the inter-program calls to sub routines and functions in a graphical form to enable an analyst to trace through the sequence of calls as well as the inter-dependency of the routines with each other in order to aid in understanding the routine operations.

### 6. **Code Module Dependency Graph**

This graph shows the intra-program dependencies, helping identify core routines from utility routines, and also trace through the programs associated with certain functions from the menu system.

### 7. **Project Estimation**

Based on the extensive information that is gathered by CodeMiner© the tool uses the Cocomo (functional points) estimation model to estimate the man hours needed for the conversion of the legacy

application. This is a ball park figure which can be used for budgetary estimates and high level planning purposes to determine the cost benefit of migration.

#### 8. Database Based Reporting Module for Data mining related Queries

Finally, all the information generated by CodeMiner© is stored in a relational database so it can be queried and mined for specific information at any point in time.

## 6 Conclusion

The key challenge facing IT organizations in migrating legacy systems is around acquiring the correct resource mix, existing legacy application resources are domain proficient but lack technical knowhow in the target platform, whereas new resources that are knowledgeable in the target platform are weak in the application application domain, necessitating a large commitment from business and IT staff.

Code converters that automatically convert one language to another do not help either because they do not convert architecture or design hence the new system inherits some of the same limitations as the old.

Visionet's legacy migration approach addresses both of these problems:

1. Visionet's approach to application migration involves application re-engineering where the original architecture is studied and a new design for the system is proposed based on modern tools and technologies available in the target platform.
2. By using CodeMiner© existing applications functionality and business rules is harvested directly from the source code and analyzed by business analysts to develop detailed functional specifications of the system in the target architecture.
3. Finally, Visionet employs a test driven development (TDD) approach to develop the application, where the system is "incubated" and validated throughout the development cycle to ensure a quality and faster delivery.