International Journal of Managing Public Sector Information & Communication Technologies (IJMPICT)
Vol.1, No.1, September 2010

Business Rule Management Framework for N-Tier E-Business Applications

M.Thirumaran, P.Dhavachelvan and S.Subhashree

Abstract

Business rules have attained a major role in the development of software systems for businesses. They influence the business behavior based on the decisions enforced upon a wide range of aspects. As the business requirements are subjected to frequent amendments, new business rules have to be evolved to reinstate the previously formed ones. Business analysts count on the assistance from the IS developers for this accomplishment. As business rules are set and owned by the business, provisions must be enforced for business rule management by the business analysts directly. In order to get a clear picture of what the current policies and terms in the business are, business users utilize the document as a valuable tool. Unfortunately, documents are not updated as per the modifications in the source code. Moreover, as the software becomes larger, documents become increasingly large and hence difficult to understand and maintain. Thus they turn out to be a non-useful resource in such conditions. With a motive to resolve the above issues, several tools for extracting the rules from the business program code have been developed. In this paper, a robust architecture for an extraction engine to isolate rules from the base source code has been proposed. This suggested model involves slicing of code segments which results in easy identification of domain variables which in turn would result in extraction of business rules, validating them and exchanging them with the newly formulated business policies. This is a new research dimension which paves waves for efficient and quick business rule management strategy.

Keywords : Business Rule Management, Business Process Model, Slicing, Segmentation, Rule Extraction

I. Introduction

Business rule is an operational rule that defines or constrains some aspect of the business that is intended to assert business structure or influence the business behavior. The tactical detail of how exactly a strategy will translate to actions is catered by the implementation of business rules. Hence the organizations of the current business world tend to proactively describe their business practices in a database of rules. Maintenance of this business rules database is a major task in each business which necessitates a lot of efforts from the software developers. Organizations might generally hire a consultant to come through to document and consolidate the various standards and methods currently in practice. Moreover, little stagnation is found in the area of business rules development in any organization. The cause for this is attributed to the growing number of changes in the business policy that are warranted every other day. In order to respond to the new business requirements, the business rules demand modifications and the software that implemented the business rules are also tailored. As a result, a variety of tools and approaches can be found today that provide IS developers with facilities for managing business rules in IS development. Business rules, however, do not pertain to the involvement of IS or to its application software. As business rules are set and owned by the business, provisions must be enforced for business rule management by the business analysts directly. Business users are therefore looking for ways to quickly edit rules to get the processes more in line with changing business needs. This effort is being intended as it turns out to be a major breakthrough in the reduction of time and manpower in large organizations.

Transformation of business policy would not be possible in the absence of a proper knowledge of the currently running business rules. The maintenance analyst must be capable of judging how an existing program can be enhanced to perform different functions or to be reused in a modified context. He has to decide as to what has to be done with a program, whether to continue the maintenance, or to reengineer it, or whether to redevelop it. In order to take this key decision, extensive information on the size, the complexity and the quality in terms of comparable numeric scales or metrics are required. His knowledge requirements are at the conceptual level as he needs to know what business functions a program is intended to carry out. Owing to this factor, business analysts tend to turn to documents for periodical recordings of the alterations made in the software program. But this approach turns out to be a failure because the documents are not maintained and updated at regular intervals as expected. The principal reason for this setback is the size of the software used in the business. In course of time and numerous update phases, as the software becomes larger, documents become increasingly large and hence difficult to understand and maintain. When the reliability of documentations is scaled down, business personnel directly look into the software codes for performing updations in the rules. At this stage, a potential tool for the extraction of the business rules from the primary source code is a basic need in the business.

Extraction of business rules from source codes demands the identification of those rules in the first place. After the rules have been identified, the validation and exchange of those rules with the modified ones would be feasible. Direct extraction of rules from the code can be attainable only by the slicing of code segments which in turn would result in domain variable recognition. These domain variables can be made to act as a trigger for the detection and modification of the business rules. This paper puts forth the development of an architecture for an extraction engine which can be employed to derive the domain variables and fetch the business rules from the source program of the software. A framework has been brought out to illustrate the working mode of the proposed engine and a sample application of the engine over a business model has been portrayed to demonstrate its potentiality.

II. Literature Survey

The people who truly know the business rules like business managers essentially implement the business rules through software developers. This may cause the business rules used in IT systems not align with organization goals and lack of agility to change. Thanawut Auechaikul presented a method that enables the business managers to create and edit the business rules themselves by constructing decision tables as a guideline and exposing them as web services in BPEL (Business Process Execution Language) to support the expected business processes [1]. Matthew L. Nelson and Robert L. Rariden examined the management of business rules management systems (BRMS) development and deployments. They revealed a business rules management lifecycle inclusive of the steps: align, capture, organize, author, distribute, test, apply and maintain [2]. Maria-Eugenia Iacob focused on business rules as a means to raise the level of abstraction (and automation) at which business logic is incorporated in model driven application design in the context of service oriented architectures. A model-driven framework for the rule-based design of services has been proposed and the extent to which, the model driven design process can be complemented and combined with business rules written in nearly natural language has been explored [3]. Xiaojian Liu introduced a new hybrid rule-based language, which integrates seamlessly with object-oriented language. This language had two distinct properties: separates business rules as a single module from the main program to facilitate the system modification and the maintenance; and allows the close interaction between the rule context and the main program context. Using this language, the complicated relationships between the objects can be expressed as a set of rules, and the rules under the help of the rule engine can be executed[4]. Marko Bajec and Marjan Krisper proposed a methodology that helped business people and developers to keep business rules at the business level inline with the rules that are implemented at the system level. In contrast to several existing approaches that primarily focus on business rules in the scope of an application, this methodology addressed the entire IS of an organization[5]. Several business process languages such as WS-BPEL have emerged for specifying business processes based on Web service technologies. This approach lacks flexibility in terms of capturing and executing the business rules that define how certain activities work and how decisions are made. Hence, VIDRE, a distributed service-oriented business rule engine, was proposed by Florian Rosenberg which enabled business processes or enterprise applications to access business rules as easily as a database by exposing them as Web services[6]. Olga Levina suggested that focusing on business rules concept for system analysis can efficiently facilitate the communication between business and informational technology - based approaches and provide a necessary framework for solving managerial problems. Business rules are considered as an instrument for systems analysis and requirements capturing. Incorporating the business rules dimension into the system analysis will help a systematic capturing of requirements and factors for a better definition and analysis of the observed system[7]. Olegas Vasilecas and Aidas Smaizys analysed business rule based data analysis for decision support and automation possibilities using business rule transformations. They proposed a specific method for business rule transformation to the executable instructions in data analysis software systems. The proposed method enabled to take the business rules used for data analysis out of software system code and use them for automated decisions and decision support to allow immediate and adequate reaction to the business environment changes according to the internal and external influences such as changes in law, new competition etc[8]. Generally large legacy systems involve large amount of code, domain variables, synonym variables and business rules, which make it more difficult to extract business rules from them. The usefulness of the methods proposed earlier to resolve this issue was limited. Zinyu Wang proposed a framework which offered distinct advantages over the normal solutions. The framework consisted of five steps:slicing program, identifying domain variables, data analysis, presenting business rules, and business validation[9]. Aqueo Kamada proposed a model for flexible business rules modeling in the context of Model Driven Architecture from OMG - Object Management Group. Considering that some business logic portions are quite volatile and susceptible to changes and other portions are quite stable and less susceptible to changes, the model can capture business changes and quickly implement them into computational systems[10]. Web service composition languages like BPEL4WS and BPML define the composition on the basis of a process that specifies the control and data flow among the services to be composed. The whole business logic underlying the composition including business policies and constraints is coded as a monolithic block. Hence, business rules are hard to change without affecting the core composition logic. Anis Charfi proposed a hybrid composition approach of breaking the composition logic into a core part (the process) and several well-modularized business rules that exist and evolve independently [11]. Andy Kellens and Kris De Schutter provided an experience report on the use of aspect-oriented technology as a means to modularize the implementation of business rules in an object-oriented, large scale case study. The goal of the refactoring of the system was to provide a proof-of-concept implementation of how such an aspect-oriented solution can improve the modularity and the extensibility of the business rule implementation [12]. Business rules are operational rules that business organizations follow to perform various activities. As the encompassing software becomes large and aged, the business rules embedded are substantial and difficult to extract. The method of using a generic tool to extract all business rules of system may be an expensive exercise due to thousands upon thousands code. A tailored solution approach to the rule extraction problem, which consists of prime program slicing, prime domain variable identifying and data analysis, rules validation was proposed by Chengliang Wang [13]. Harry M. Sneed & Katalin Erdos reviewed the state of the art on application knowledge acquisiton from existing software systems and defines the role of business rules. They also presented a method for identifying and extracting business rules by means of data output identification and program stripping. The method was implemented in a reverse engineering tool SOFT-REDOC for COBOL programs [14]. Companies follow the business rules to conduct their businesses daily where the business rules are often implemented into the software systems. Chia-Chu Chiang proposed a method for identifying and extracting business rules from legacy code by means of data identification and program slicing. The extracted code corresponding to a business rule is then compiled into a component for interoperability where the component is built conforming to a protocol. The components abiding by the protocol can communicate with each other seamlessly in a heterogeneous object oriented computing environment [15]. It is common to imbed business rules within the code of distributed object systems. When business practices and policies change it is difficult to correctly reflect those changes in the applications implementing them. Isabelle Rouvellou described a framework that enabled enterprises to develop distributed business applications that systematically externalize the time- and situation-variable parts of their business logic as externally applied entities called business rules [16]. Web services composition process needs business rules to regulate the behavior of the partner services. However, designing these rules is time-consuming and error-prone. In this spirit, rule analysis and verification for services composition is urgently required to augment its reliability and usability. Hai Liu and Qing Li considered a variant of Description Logics(DLs), called ALCO(Q*) as the underlying logic, and provided a formal mapping to transform ECA rules, so that the semantics in the original ECA rules can be captured and are computationally traceable [17].

III. Type of Business Rule

A Structural assertion a defined concept or a statement of a fact that expresses some aspect of the structure of an enterprise. This encompasses both terms and the facts assembled from these terms. An Action assertion is a statement of a constraint or condition that limits or controls, the actions of the enterprise. A derivation a statement of knowledge that is derived from other knowledge in the business. The Business rules is better understood, techniques and tools will be developed to support the missing elements of the task. Techniques would include formal methods for describing rules rigorously, with tools translating these formalisms directly into program code or other implementation constructs. While graphic techniques have existed for some time to describe data structure and functions, formal methods for describing rules are relatively new and still being refined.

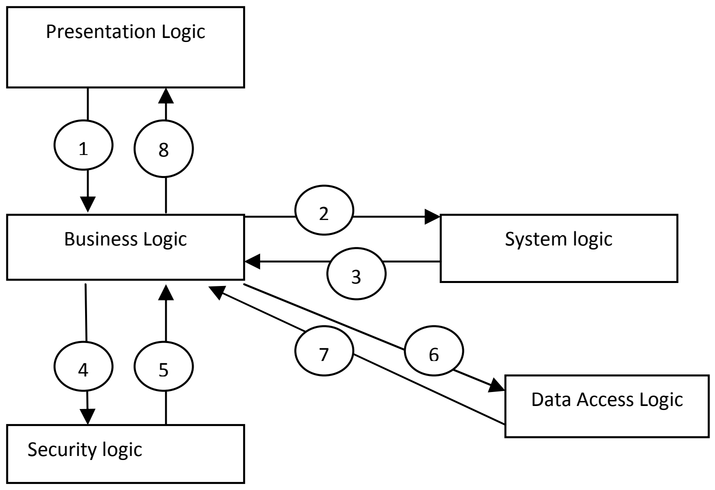


Figure 1 . Enterprise E-Business application

Enterprise E-Business System is classified into number of tiers such as business logic, data access logic, security logic, etc. The above diagram shows various tiers of E-business system and their interaction. Enterprise E-Business application outlines the goals and computer of customer relationship management, enterprise resource planning, and supply chain management, and discusses the benefits and challenge of this e-business application. The presentations Logical view because you can choose the way the documents are displayed on your system. Business logic, or domain logic, is a non-technical term generally used to describe the functional algorithms that handle information exchange between a database and a user interface. It is distinguished from input and output data validation and product logic. Models real life business objects, prescribes how business objects interact with one another. Enforces the routes and the methods by which business objects are accessed and updated. Security Logical consists of software safeguards for an organization's systems, including user Identification and password access, authentication, access rights and authority levels. These measures are to ensure that only authorized users are able to perform actions or access information in a network or a workstation. It is a subset of computer security. A Data Access Logic (DAL) is a layer of a computer program which provides simplified access to data stored in persistent storage of some kind, such as an entity-relational database. The DAL might return a reference to an complete with its attributes instead of a row of fields from a database table. This allows the client (or user) modules to be created with a higher level of abstraction. This kind of model could be implemented by creating a class of data access methods that directly reference a corresponding set of database stored procedures. The DAL hides this complexity of the underlying data store from the external world.

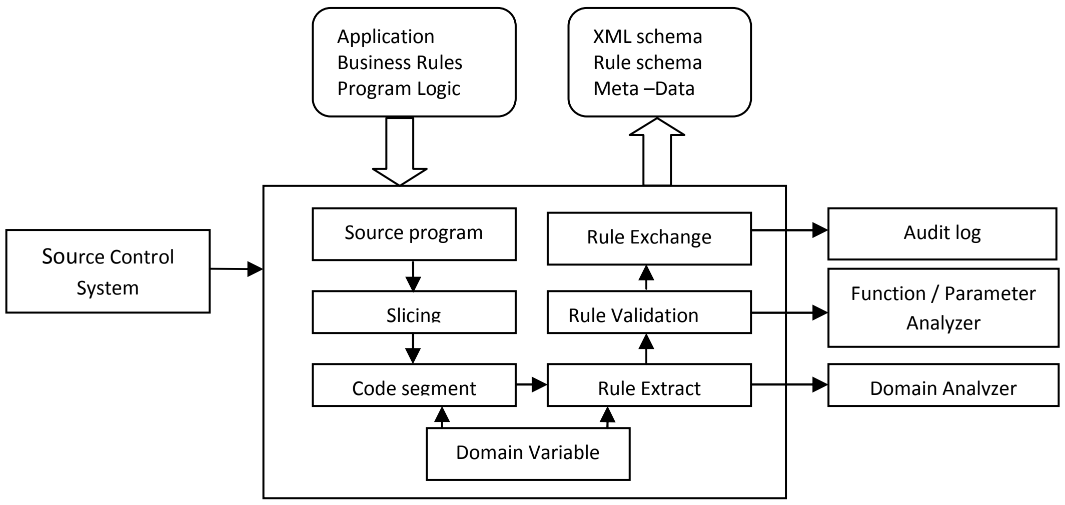


Figure 2. Architecture of Business Rule Extraction Engine

Business Rule extraction engine gets the source program as input and perform segmentation and slicing over the source input in order to identify the rules by encountering domain variables. Program slicing is a technique for simplifying programs by focusing on selected aspects of semantics. The process of slicing deletes those parts of the program which can be determined to have no effect upon the semantics of interest. Slicing has applications in testing and debugging, re-engineering, program comprehension and software measurement.

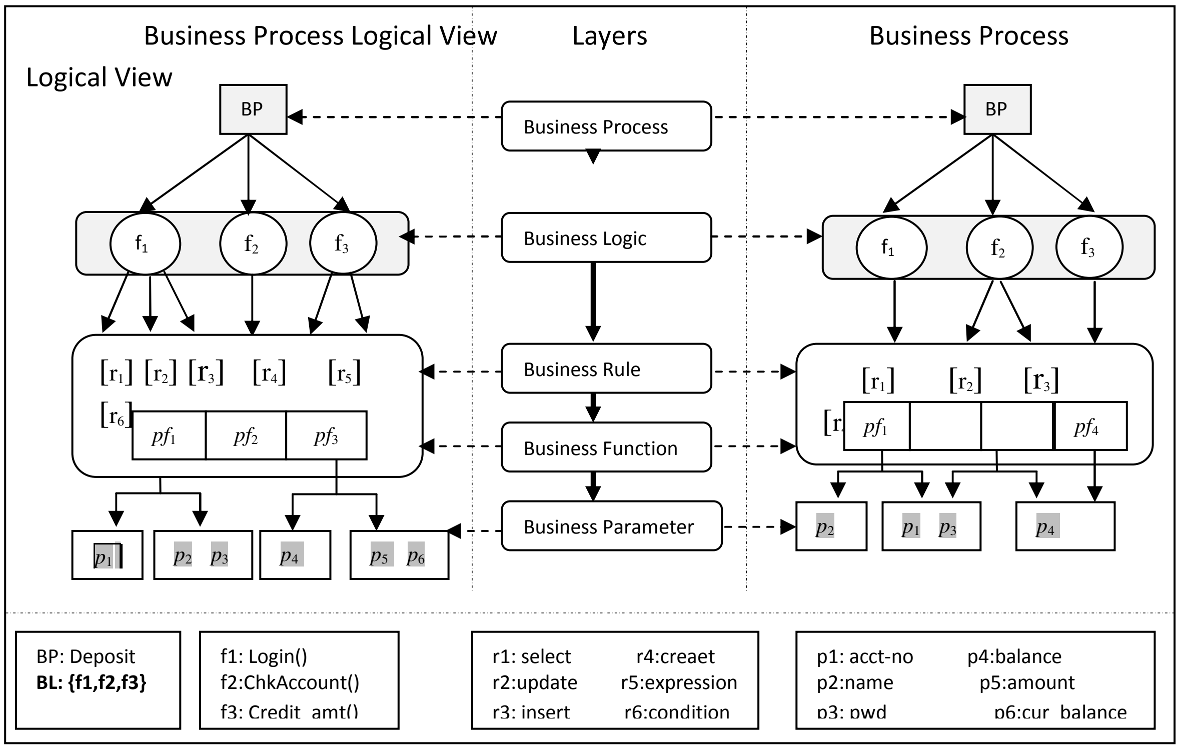


Figure 3. Working Model of Business Rule Analysis

// Deposit Business Process contains logic which does three subsequent tasks as listed below

f1 : login verifiaction by username and password

f2 : Account Verfication

f3: Deposit the amount

BP= {f1,f2,f3}

r1:get username and password

r2:select username and password from the datastore

r3 : match username and password

f1=[r1][r2][r3]

r4: Get Accountdetails with reference to [r1,r2,r3]

f2 =[r4]

r5: Get old balance through [r4]

r6: Get amount to be credited and add it with old balance

f3=[r5][r6]

(a) Algorithm for Parsing

```
Algorithm Parsing(ProjectSolution)
Input: ProjectFiles,ResourceFile,ObjectFile,Solution
Output : ClassList,ObjectList, MethodList,ParameterList
Begin
//Lexical analyzer to scan and tokenize the input source program
Procedure sourcefilter(code)
Begin
New_src→ Remove_Header(code)
New_src→Remove_Namespace(code)
New_src→Remove_comments(code)
Return new_src
End    // end of sourcefilter
Procedure Tokenizer(new_src)
Begin
While(new_src != EOF)
begin
src→Readline(new_src)
TokenList→Lex(src)
Return TokenList
End // end of tokenizer
//------------------------------------------------------------------------------
// Grammar rules to parse the procedure and function
Source→<StmtList>
StmtList→<stmt>|stmt <StmtList>
Stmt→<procedure> |<function>
Procedure → <return_type> procedure <proc_name>
  (<paramlist>)
Function → <return_type> function    <func_name>    (<paramlist
Return_type→ void |float |int | string
Paramlist→<param> | param , <paramlist>
//------------------------------------------------------------------------------
// identify and store the mehtods in an array list MethodList[n]
// update the domain variable list DomainVariables[n]
// LR_Parsing(stmtlist,grammar)
Procedure classifier(TokenList, Tokentype)
begin
While(StmtList.next())
begin
If TokenList(startswith( procedure) or startswith( function) ) then
Begin
//array of methods
Add    proc_name to MethodList[n]
Add func_name to MethodList[n]
// array of domain variables
Add param to DomainVariables[n]
Else
If TokenList(startswith( class) then
Add class to ClassList[n]
End // end of if
End// end of while
End// end of classifier
End //end of main
```

(b) Algorithm for Slicing

```
Algorithm
  Slicing(Source,ClassList,MethodLit,DomainVariables)
Input: ProjectFiles,ResourceFile,ClassList,MethodList
Output : ClassList,ObjectList, MethodList,ParameterList
Begin
While(StmtList.next()==TRUE)
Begin
Stmt→Getinput(StmtList)
TokenList →Call Tokenizer(stmt)
// tokenlist matches with domain variables
If TokenList[i] ε [domainVariables] then
// if tokenlist is procedure or function then find the end of
  procedure
//And add each stmt into the slice
If TokenList(startswith( procedure) or startswith( function)
  and
(TokenList(endswith(end) ) ) then
begin
i=0
// Append stmt into the slice if the stmt_type is expression or
  condition or loop or input_output_stmt.
For each stmt in StmtList
begin
If ( stmt_type = expression ||stmt_type=condition ||
Stmt_type=loop || stmt_type=input_output_stmt) then
Stmt is valid
Slice[i] →Append_stmt(stmt)
i=i+1
end //for
end    //if
End    //end of main
```

```
Algorithm RuleExtraction(Slice, Domainvariables)
Input: Program Slice, Domain Variables
Output: RuleSet, Audit_log
Begin
// get each stmt in a slice and tokenize it then validate it for
  DML
//Operations such as select, insert, update and delete, if so then
//Add the stmt into RuleSet
//RuleSet consists of DML queries
For each stmt in Slice
TokenList →Call Tokenizer(stmt)
If TokenList has DomainVariables then
If ( stmt_type = select ||stmt_type=insert ||
Stmt_type=update || stmt_type=delete) then
IsQuery=TRUE
Query→Call getQuery(stmt)
RuleSet[i]=Query
AttributeList→ getAttribute(Query)
Table_name→FindTable(Query)
WhereClause → FindCondition()
```

Business Rule Management Framework used to create, manage, and support business rules of an enterprise. They bridge the gap between business and IT by giving the control of business logic to business analysts. The basic idea is that the BRMF decouples an application's business logic from its data validation logic and from its flow control. The business logic runs in its own container, the "Rules Engine," and business analysts code the business rules in a simple English-like programming language, which is very simple to understand and learn. Even a business user who has never programmed in any programming language can easily create and manage these rules [13].

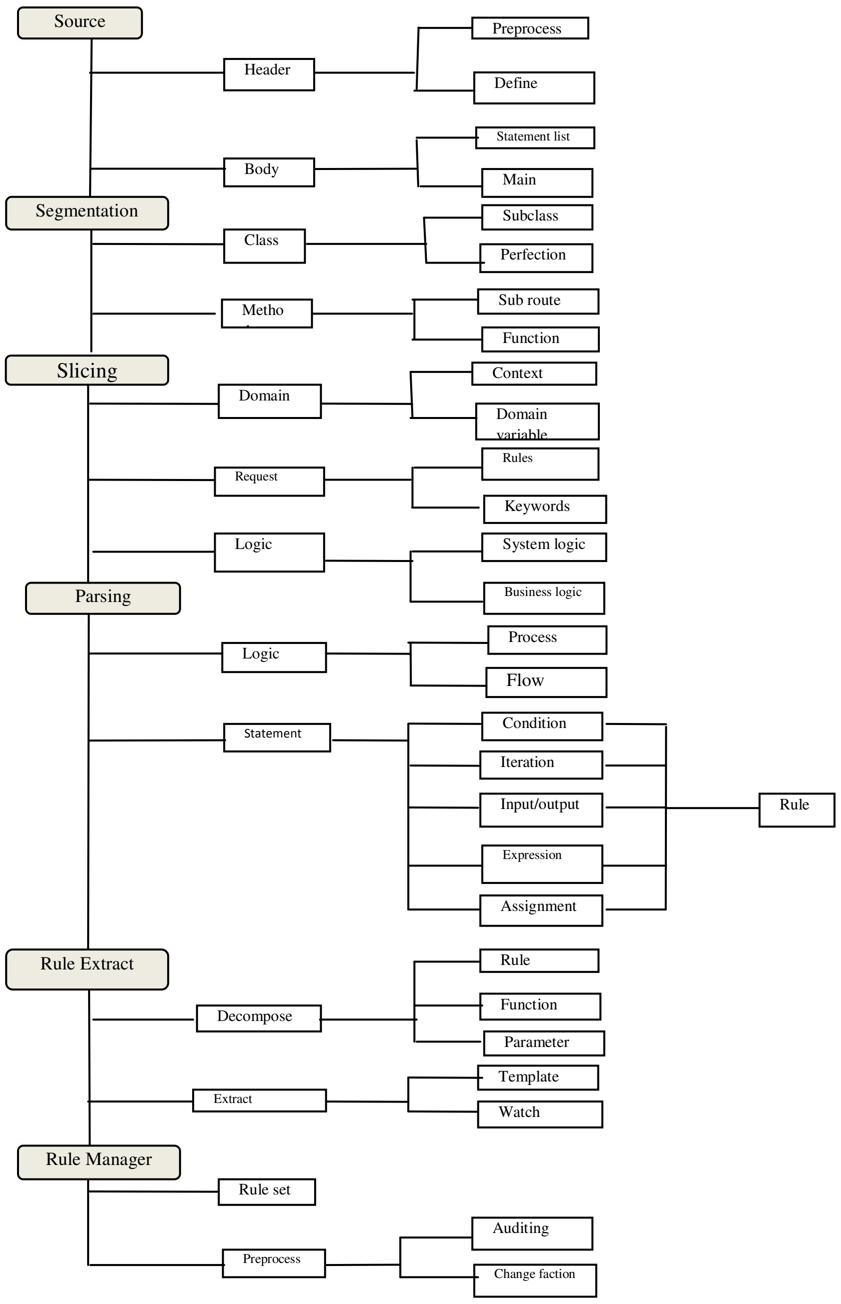
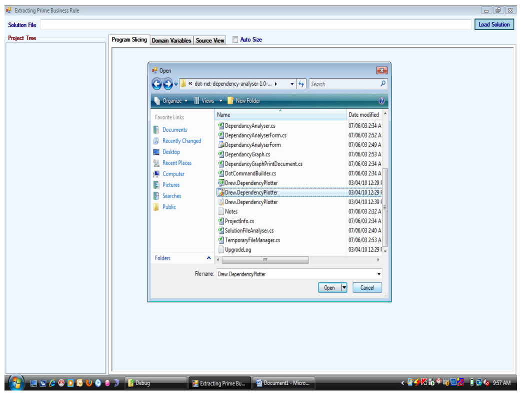


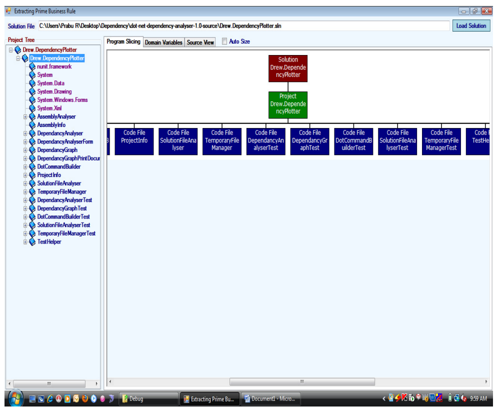
Figure 4. Business Rule Analysis and Extraction Phases

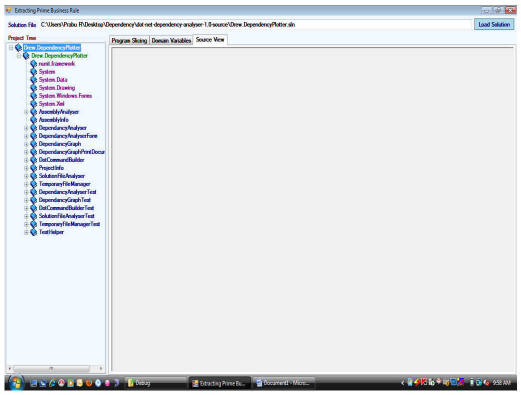
Parsing is done to analyze a given input sequence to determine its grammatical structure with respect to a given formal grammar. The source statements are syntactically analyzed to categorize them as to which type of statements they belong to. This is done by dividing the code into various functional components. Program slicing is a proficient method used by experienced computer programmers for abstracting from programs. Starting from a subset of a program's behavior, slicing reduces that program to a minimal form which still reflects that behavior. The reduced program, called a "slice", is an independent program assured to faithfully represent the original program within the domain of the specified subset of behavior. Program slicing is an efficient method used in numerous applications like debugging, software maintenance, optimization, program analysis, and information flow control. Here parsing and slicing are employed to recognize the domain variables and hence extract the business rules. The algorithm for parsing and program source slicing are given above.

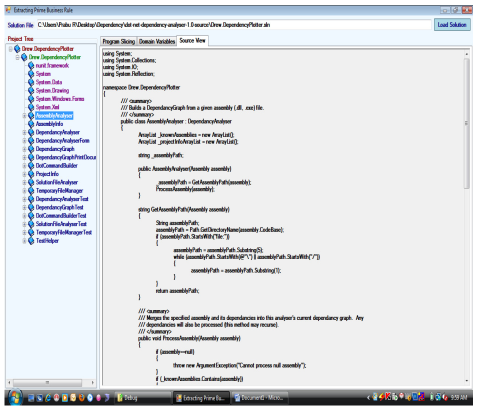
I. After Parsing Solution File



II. Input Screen- Load Solution File







```
using System;
using System.Collections;
using System.IO;
using System.Reflection;

namespace Drew.DependencyPlotter
{
        /// <summary>
        /// Builds a DependancyGraph from a given assembly (.dll, .exe) file.
        /// </summary>
        public class AssemblyAnalyser : DependancyAnalyser
        {
                ArrayList _knownAssemblies = new ArrayList();
                ArrayList _projectInfoArrayList = new ArrayList();

                string _assemblyPath;

                public AssemblyAnalyser(Assembly assembly)
                {
                        _assemblyPath = GetAssemblyPath(assembly);
                        ProcessAssembly(assembly);
                }

                string GetAssemblyPath(Assembly assembly)
                {
                        String assemblyPath;
                        assemblyPath = Path.GetDirectoryName(assembly.CodeBase);
                        if (assemblyPath.StartsWith("file:"))
                        {
                                assemblyPath = assemblyPath.Substring(5);
                                while (assemblyPath.StartsWith(@"\") || assemblyPath.StartsWith("/"))
                                {
                                        assemblyPath = assemblyPath.Substring(1);
                                }
                        }
                        return assemblyPath;
                }

                /// <summary>
                /// Merges the specified assembly and its dependancies into this analyser's current dependancy graph.  Any
                /// dependancies will also be processed (this method may recurse).
                /// </summary>
                public void ProcessAssembly(Assembly assembly)
                {
                        if (assembly==null)
                        {
                                throw new ArgumentException("Cannot process null assembly");
                        }
                        if (_knownAssemblies.Contains(assembly))
```

III. Program Slicing Output

IV. Code Parsing for Domain Variable Identification

We have implemented the above discussed algorithms in C#.NET and the proposed system successfully generates the parsing results after slicing. After parsing though domain variable identification the identifies business rule is extracted for further modification or rule replacement enforced by the business expert in the effect of compliance or new business demand.

Conclusion

Rules are most often embedded in operational software systems for business. In this paper we have presented a framework for business rule management of N-Tier Business applications. Though methods for rule extraction from large systems have been proposed earlier, this engine has been designed to capture, model, store, implement, test, and execute business rules in a profound and efficient manner. This Business Rules Management Framework platform creates an abstraction layer between the application code and the business logic. This allows the manipulation of business rules independent of the application code. The architecture of the proposed engine has been described and a mathematical model has been considered for this purpose. The architecture is also evaluated using BPPEE model and its application in slicing the business codes into code has been illustrated. Model level program slicing followed by domain variable derivation is supplemented by data analysis and rule validation in order to completely exchange the previously formulated conditions with the newly enforced rules. This architecture for n-tier e-business application system needs to be reengineered with new technology for performance enhancement.

References

[1] Thanawut Auechaikul and Wiwat Vatanawood "A Development of Business Rules with Decision Table for Business Process", proceeding of IEEE Region 10 Conference 2007.pp1-4.
[2] Matthew L. Nelson, and Robert L. Rariden, Ravi Sen, "A Lifecycle Approach towards Business rules Management", Proceedings of the 41st Hawaii International Conference on System Sciences – 2008.pp 113-113.
[3] Maria-Eugenia Iacob, Henk Jonkers "A Model-driven Perspective on the Rule-based Specification of Services", Proceeding of 12th International IEEE Enterprise Distributed Object Computing Conference, 2008.pp75-84.
[4] Xiaojian Liu, Xuejun Liu and Jianxin Li "A Hybrid Language Combining Business Rules with Object –Orientation", proceeding of IEEE International Conference on 2008.
[5] M. Bajec and M. Krisper "A Methodology and tool support for managing business rules in organization", Information System 30 (2005), 2004. pp 423-443,

[6] Florian Rosenberg, Christoph Nagl, Schahram Dustdar "Applying Distributed Business Rules- The VIDRE Approach", IEEE International Conference on Services Computing (SCC'06), 2006. pp 471-478.

[7] Olga Levina "Bearing the Challenge of Multidisciplinary by Business rules based system Analysis", proceeding of IEEE Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on 2009. pp 447- 452.

[8] Olegas Vasilecas, Aidas Smaizys "Business rule based data analysis for decision support and automation" International Conference on Computer Systems and Technologies CompSysTech'06.

[9] Xinyu Wang, Jianling Sun, Xiaohu Yang, Zhijiun He, Srini maddmeni, "Business Rules Extraction from Large Legacy System", Proceeding of IEEE 8th Europen Conference on Software Maintenance and Reengineering (CSM'04), Tamper, Finland, March 24-26, 2004,pp 249-254.

[10] Aqueo Kamada "Business Using Rules and services in the context of model Driven Architecture", Proceeding of 11th IEEE International Conference Science and Engineering, 2008.

[11] Anis Charfi, Mira Mezini "Hybrid Web Service Composition Business process Meet Rules", ICSOC'04, November 15–19, 2004.

[12] Andy Kellens, Kris De Schutter, Theo D'Hondt and Viviane Jonckers "Experience in Modularizing Business rules into aspect", Proceeding of IEEE international conference on ICSM 2008.

[13] Chengliang Wang, Yaxin Zhou, Juanjuan Chen, "Extracting Prime business rules from legacy system" Proceeding of Computer Science Software Engineering, Proceeding of International Conference on Computer science on 12-14Dec.2008,vol-2, pp 19-23.

[14] Harry M. Sneed and Katalin Erdos, "Extracting Business Rules from Source Code", Proceeding of IEEE 4th International Workshop on Program Comprehension(IWPC'96), 1996. pp 240-247.

[15] Chia-Chu Chiang "Extracting Business Rules from Legacy System into Reusable components", Proceeding of the 2006 IEEE/SMC International Conference System Engineering. 24-26 April 2006. pp 6.

[16] Isabelle Rouvellou, Kevin Ramus, "Extending Business objects with Business Rules", proceeding of IEEE 33rd international Conference on 2000,pp 238-249.

[17] Hai Liu, Qing Li, Naijie Gu ,and An Liu "Exploiting Semantics for Analyzing and Verifying Business Rules in web services Composition and Contracting", IEEE International Conference on Web Services2008.