

## Assessment on Distributed Collaborative Applications Research and Reengineering

Cosmin TOMOZEI

Department of Mathematics and Computer Science,  
“Vasile Alecsandri” University of Bacău,  
cosmin.tomozei@ub.ro

Bogdan PĂTRUT

Department of Mathematics and Computer Science,  
“Vasile Alecsandri” University of Bacău,  
bogdan@edusoft.ro

***Abstract:** The objective of this paper is to reflect about the research results achieved in distributed collaborative applications reengineering. Collaborative software proves to be a key-element in achieving common goals in many areas of activity, such as education or healthcare. On the other hand, reengineering provides efficiency, robustness and reliability, offering the possibility for software applications to be utilized for longer periods of time with better results.*

***Keywords:** collaborative applications, software reengineering, semantics, educational portals, electronic journals.*

### 1. Semantic reengineering of distributed collaborative applications

Collaborative systems appear as entities which bring the possibility of computer supported collaborative work, applied in many areas of activity. Computer supported collaborative work is an activity realized by people from remote places, with a common objective. Research regarding the collaborative work is made, in order to study their interactions and the coordination in virtual teams.

Distributed systems offer the possibility for collaborative applications to be created, utilized and maintained efficiently. We cannot imagine collaborative software architectures without having in consideration the Internet. Maintaining the systems is a must, as well as keeping them valuable for longer periods of time with reasonable costs. Efficiency presumes in this case reengineering, from the both sides, semantic and syntactic.

Reengineering appears as a consequence of the necessity to improve the quality of the results. The objectives determined by the need of reengineering are somewhat diverse, but do not differ significantly from the existent ones.

This process offers new functionalities, new objectives or higher levels of quality, reflected in metrics.

Semantic reengineering defined in [1] is not an unconnected process. It includes other arguments and needs them to complete the tasks. Language semantics is very important in information exchange as well as data semantics and code semantics. In consequence, when defining semantic reengineering, we also take into account the transformations in data, language and code.

The main idea is to develop structural, functional and architectural transformations with minimum efforts with the result of restructuring the entities architecture, design and source code in order to bring evolution in the meaning.

In [1] we have defined as well the reengineering and development functions. Their product reflects the advances in quality of the results provided by reengineering.

We would like point out that syntax must subordinate itself to semantics, because *the meaning* is the most important. Syntactic modification is only a way for bringing evolution in semantics.

Reengineering in operational semantics comes as a transformation or conversion of mathematic formulae in source code.

Operational semantics is stalwartly related with the transition of a software system.

In [2] system transition is defined as a set of configurations and a binary defined relation which brings the system from an initial configuration ( $\beta_0$ ) to a final one ( $\beta_1$ ).

We have to consider each configuration related with a specific meaning which is not considerably dissimilar from the previous one. Transformation from the initial to a final step, which will bring a qualitative growth in the meaning and a better realization of the objectives, is defined as *semantic reengineering* [1].

The process of semantic reengineering is finite, deterministic, value orientated and progressive. The following elements will be more explanative:

- *finiteness* which means that reengineering is considered to have a finite limit and will be finalized after realizing all the transformations;
- *determinism*, due to the necessity of doing a number of iterations until the process is finalized and the meaning of the software faced a new qualitative upward;
- *continuity* during the development cycle;
- *flexibility* regarding the development platforms;
- *appropriate management* which guides the development team in the process.

The area in which the meanings of the programming languages are used through the mathematical objects construction that describes them is defined as *Denotational semantics*. Since it is related with the states, in which the commands are partial functions of the states domains, denotational semantics is very important as denotation of data structures and types, such as trees, vectors or graphs.

New models for concurrent computatoion, for instance the actor model or Petri nets has been introduced by denotational semantics. From the denotational point of view the translation of the source code is also taken into discussion. If we want to take into account the web application functions, semantic reengineering and denotational semantics may be used to turn it from one language into another.

In order to make transformations in software code and to make specifications either syntactically and semantically we may use web orientation of software applications. SOAP applications [3], also named web services use web methods just as traditional software implements functions and procedures. Interoperability is provided by reengineering, which will adapt the software code, in such a way as to offer it possibilities to operate more general problems.

Transformation by reengineering produces the following piece of code as web method with the result of an XML, understood by software applications which communicate over the Internet. The clients will retrieve information from the relational database just by referring and invoking web methods.

## **2. Preliminary aspects regarding distributed collaborative systems development**

Software engineering and reengineering are considered to be actual and important in the industry, an increased number of specialists being involved in researching about this

topic. The results of their research are both theoretical and methodological. Practical work from software companies, laboratories and universities is also not to be ignored and provides valuable scientific results about the theme.

This section is focused mainly on describing the way we reengineer distributed collaborative systems, in order to obtain a higher level of efficiency and quality in the results.

Due to the increasing level of software systems complexity, it is practically impossible to start the development process from green field over and over again. Fast changing of the objectives which have to be accomplished by computer programs and IT systems in general presume that *maintainability* must become the most significant metric. In the same time, when talking about dimensions, volume of operations, or necessity of computing large volumes of information, the *distribution* of software, hardware and data come as a logical consequence. Maintaining an adequate level of integrity and security involve distribution as well, hence being subjected to reengineering.

In [4] there are presented some aspects regarding distributed systems that we mention in the following:

- *multiple nodes*, connected in a computer network; distributed systems suppose that multiple computers are connected and share tasks in order to realize the objectives; parallel processing is not to be confused with distributed computing; while parallel processing involves many processors on the same machine, distribution means at the outset task sharing;
- *concurrency*; each node has independent functionalities apart from the other nodes and operates concurrently with the other nodes; it is likely to exist many processes on each node, and on each process there are multiple threads;
- *message passing* through protocols, such as TCP/IP over modems or Ethernet;
- *heterogeneity* of nodes; each node being dissimilar regarding hardware and software;
- *multiple protocols*, mainly asynchronous;
- *openness*; in comparison to sequential programs, which are mainly closed and do not change their configuration during execution, in distributed systems we can add nodes while the whole ensemble is functioning; the openness presumes that each node satisfies a set of conditions and protocols to ensure *interoperability* with the components added or modified;
- *fault tolerance and transparency* allow users to cooperate with the software system without knowing if there are components that don't work in a certain moment of time; this fact should not affect the general functioning of the system;
- *persistence*; data is stored in a persistent, non volatile environment, such as databases, data warehouses and storage servers;
- *security*; each user should interact with the system according to the rights he has; elements such as *authentication servers*, *firewalls*, *antivirus* technologies are to be used;
- there is *no central server*, but there are multiple servers which cooperate in order to achieve nonstop, uninterrupted operation of the system.

Distributed applications consist of the software components that run on the distributed systems, and have the following characteristics:

- they are built in distinct development environments, they operate in diverse environments, on different operating systems, on platforms that are connected in a computer network;
- they are built on two tiers (client - server), three tiers (client - middleware - server) or multitier (client - multiple middleware - multiple servers).

In [5] we made the following remarks regarding distributed applications as well.

In *client - server applications* each side, *client* or *server* are referred to as nodes on the computer network. The client sends requests to the server, can be connected to a reduced number of servers and offers to the user information by the mean of a graphic interface.

In *three - tier applications*, the fragments are being distributed on three levels. The first level is the *presentation* level, which can be seen from outside, the second level is commonly known as *business logic*, and the third is the *database tier*. In *N - Tier applications* consisting of a generalization of the three-tier model where an application is executed by more distinct software agents

Software reengineering offers reliability and efficiency to the distributed systems development process and provides also the possibility of reducing the duration between the appearance of a new objective or demand and the implementation of the demand in the system.

In the following sections we will talk about the distributed applications development necessity of implementing software reengineering.

### **3. F# Modules Integration and Reengineering Based on Transformation**

In this section we focus on the integration of functional modules written in F# in Windows 2008 server nodes, as parts of collaborative systems. The process of software reengineering is entitled to add new software modules in distributed environments. In the following source code, we will put into practice a web service which returns also an ADO.NET dataset which was filled with data from the database.

In figure 1 we present the functional architecture model of distributed reengineering. The process starts iteratively from the initial objectives which are realized by the current software application. In the earlier stages, the application had been subjected to *forward engineering*, which implemented the abstract requirements and objectives into actual software modules.

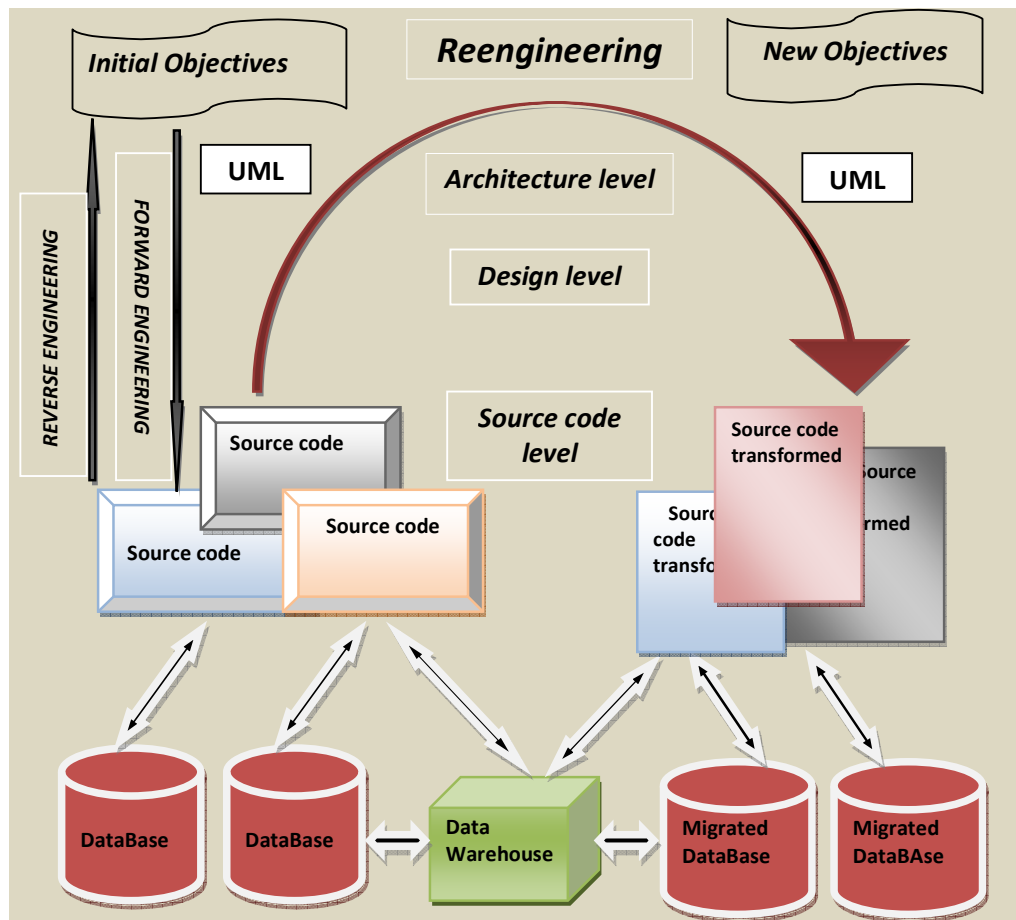


Fig. 1. Functional architecture of distributed reengineering

Forward engineering [6] consists of moving from the high level of abstraction levels of architecture and design, which are implementation independent to the physical implementation of the system. By forward engineering, we will get the result of source code and database implementations.

On the other hand, reverse engineering presumes the turn round way, from the existing implementations to the abstract architectures. Reengineering involves both of the processes, forward and reverse engineering.

Functional languages written modules, such as F# modules are to be integrated due to reengineering. In [7], we have mentioned that F# is appropriate for distributed computing, because of the power and robustness of functional programming languages and the offering the possibility of solving in a scientific conduct a set of very difficult problems.

The contribution of this new programming language to Computer Science research is very significant, joining together the simplicity of a functional language with the power, robustness and generality of the .NET framework.

The following source code presents a module written in F#, which represents a query from the same database as the above mentioned web services. It is straightforward to see that the entire software application is enriched with interoperability.

```
#light
open System.Collections.Generic
open System.Data
open System.Data.SqlClient
```

```

open System
let connString1 = "Data Source=PROGRAMARE01\SQLEXPRESS;Initial
Catalog=authorsBRAIN;Integrated Security=True;"
let fraza_sql = "select * from authors"
let connectivity = new SqlConnection(connString1)
using (connectivity)
    (fun connection ->
        let command = connection.CreateCommand()
        command.CommandText <- fraza_sql
        command.CommandText <- fraza_sql
        command.CommandType <- CommandType.Text;
        connection.Open()
        using (command.ExecuteReader())
            (fun reader ->
                let id_utiliz = reader.GetOrdinal("id_author")
                let nume = reader.GetOrdinal("article_title")
                let prenume = reader.GetOrdinal("author_name")
                let paper_name=
reader.GetOrdinal("paper_name")
                while reader.Read() do
                    Console.WriteLine (id_author)
                    Console.WriteLine (article_title)
                    Console.WriteLine (author_name)
                    Console.WriteLine (paper_name)
                ))
    )

```

To conclude, we may state that working with many database management systems and with several programming languages is a key issue in distributed collaborative systems. Integrating distinct functional modules could help in that direction. In collaborative project development, functional language build modules are to be taken into consideration.

#### 4. Practical implementations in education and research

The BRAIN initiative is defined as a collaborative project from two sides: the development perspective and the editorial and administrative perspective. The main purpose of this journal is to bring specialists from many areas of activity in collaborating and making valuable papers.

Starting from the idea that everything is related to the brain, we expect from the authors all kind of scientific papers in different fields, which are related with the brain. The topics are very different, but the study of the functions of the human brain and possibilities to create artificial intelligence is the main idea and the central goal.

The aim of this journal is to create links between researchers from apparently different scientific fields, such as Computer Science and Neurology. In fact, there are a lot of topics such as Artificial Intelligence, Cognitive Sciences and Neurosciences that can intersect in the study of the brain and its intelligent functions.

Our journal contains (in section BRAINovations) peer review articles. These articles should be original unpublished articles of the authors. The peer review process is a blind one. The reviewers are well recognized scientists who make part of our scientific board, and independent reviewers.

Some innovative young researchers from the "Vasile Alecsandri" University of Bacau, Romania, had the idea to edit and publish the BRAIN journal in order to make an agora of interdisciplinary study of the brain. Young scientists and seniors, from artificial intelligence, cognitive sciences and neurology fields are expected to publish their original works in our journal.

The screenshot displays the BRAIN journal website. At the top, the journal's logo features three colored squares (yellow, red, and dark red) followed by the text "BRAIN" in large red letters and "Broad Research in Artificial Intelligence and Neuroscience" in smaller dark red letters. To the right, the ISSN number "ISSN: 2067-3957" is displayed. Below the header is a navigation menu with tabs for "BRAIN HOME", "CURRENT ISSUE" (which is highlighted), "BRAIN VOLUMES", "EDITORIAL BOARD", and "AUTHORS". A search bar is located to the right of the navigation menu. Below the navigation menu, the breadcrumb trail reads "You are here: CURRENT ISSUE > BRAINInitiative". On the right side of this section, there are links for "Register" and "Login". The main content area is divided into two columns. The left column is titled "Content of the current issue" and contains a list of links: "BRAINInitiative", "BRAINews", "BRAINstorming", "BRAINovations", "Lecture BRAINotes", and "WEBrain". The right column is titled "About the BRAInitiative" and contains three paragraphs of text. The first paragraph states the journal's aim to create links between researchers from different scientific fields. The second paragraph describes the journal's content, including peer-reviewed articles. The third paragraph repeats the journal's origin story. Below this is a "Call for papers" section, which includes a call to action for the next issue, a welcome message for contributions, a deadline of January 10, 2010, and an email address for submissions: bogdan@edusoft.ro. At the bottom of the page, there is a footer with navigation links, a "Privacy Statement" link, a "Terms Of Use" link, and a copyright notice: "Copyright 2009 by BRAIN".

As we mentioned above, virtual collaboration from distinct areas of activity is intended to be made, social interactions and exchange of ideas providing such possibilities.

BRAIN as collaborative project and collaborative work permit that users have *social interactions*, by using social conventions and rules, in order to communicate and build virtual teams. Social computing presumes that any kind of social behavior is supported by the computer system. In addition to social computing, social information processing is an activity that describes the way in which the human knowledge is organized by collective human actions. In the medical domain, such as surgery or neurology knowledge exchange and common experience allow to specialists from remote places to cooperate and reduce the duration of procedures, by working together for helping the same patient concurrently.

Technically, the BRAIN project is an important component of a larger research project which started one year ago [8]. BRAIN has been added by reengineering and transforming the research and education platform of the department.

We expect for the project to gain more on the collaborative plane, involving researchers from universities and laboratories from the country and from abroad.

## 5. Conclusions

This paper presented in a succinct way the results obtained by the authors, as members of the Department of Mathematics and Computer Science, at “Vasile Alecsandri” University of Bacău, regarding the collaborative systems area.

The scientific results are both from the collaborative and technical manners. The research platform from our department involves artificial intelligence applications, semantic web, distributed applications which were subjected to reengineering and functional languages integration in distributed systems.

New modules have been added on the platform, BRAIN being one of them and higher levels of quality, reflected in metrics have been achieved.

## References

- [1] C. Tomozei, „Hypertext Entities Semantic Web-Oriented Reengineering,” *Journal of Applied Quantitative Methods*, Vol. 3, No. 1, 2008, pp. 9-19.
- [2] G. D. Plotkin, *A Structural Approach to Operational Semantics*, Computer Science Dept., Aarhus University, 1981.
- [3] <http://www.w3.org/2001/sw/>
- [4] E. D. Reilly, *Concise Encyclopedia of Computer Science*, Wiley, 2008.
- [5] C. Tomozei, „N-Tier Distributed Applications Dependable Construction,” *Journal of Information Technology & Communication Security, SECITC*, 2008, pp. 65-71.
- [6] S. Demeyer, S. Ducasse and O. Nierstrasz, *Object-Oriented Reengineering Patterns*, Elsevier Science, Square Bracket Associates, 2008.
- [7] C. Tomozei, “Quality Characteristics of Business - Oriented Open Source Community Projects,” *Open Source Science Journal*, Vol. 1, No. 1, 2009, pp. 179 - 188.
- [8] M. Vetrici and C. Tomozei, “Distributed Collaborative Software Development Process Improvement Using Criticality Analysis,” *Economy Informatics*, Vol. 9, No. 1, 2009, pp. 71-78.

**Authors**

**Cosmin TOMOZEI** is University Assistant - Lecturer at Mathematics and Computer Science Department from Faculty of Sciences of the University of Bacau. He is a PhD candidate from October 2007 at Economic Informatics Department from University of Economics, Bucharest. He holds a Master in Science - Databases- Business Support from University of Economics, Bucharest. He graduated in Economic Informatics at Faculty of Economic Cybernetics, Statistics and Informatics in 2006. His main research areas are: object oriented programming, functional programming in Lisp and F#, software reengineering and distributed applications development.



**Bogdan PĂTRUȚ** (b. June 16, 1969) received his BSc in Informatics (1994), MSc in Distributed Programming (1996), PhD in Accounting and Business Information Systems (2007) from "Al. I. Cuza" University of Iasi, Romania, and PhD in Informatics (2008) from "Babes-Bolyai" University of Cluj-Napoca. Now he is associate professor of informatics at Mathematics and Computer Science Department, Faculty of Sciences, "V. Alecsandri" University of Bacau, Romania. His current research interests include diferent aspects of Artificial Intelligence. He has (co-)authored 23 books and more than 20 papers, more than 10 conferences participation, member in International Program Committee of 4 conferences and workshops.