# Distribute Service Components into Multi-level Architecture from Legacy Applications

Juefeng Li, Xiaohu Yang, Yongwei Ding, Xiaochun Zhu
*College of Computer Science & Technology, Zhejiang University, Hangzhou, P.R.China*
*{lijuefeng, yangxh, smile_ding, xzhu}@ zju.edu.cn*

## Abstract

*With the growth of organizations, the scope and size of their information systems are getting bigger. The complexity of the business logic makes the legacy system low maintainability. By providing loose coupling, Service Oriented Architecture (SOA) is becoming dominant enterprise IT architecture. How to identify reusable components from legacy system in future SOA system is becoming a hot research area. This paper proposes a multi-level architecture to meet the requirements of data separation in special organizations. Clustering technologies used to analyze and build components successfully are applied with component connectivity strength metric to distribute data and their relevant service components into levels for achieving SOA high performance, as well as flexibility and interoperation.*

## 1. Introduction

With the growth of organization, the scope and size of their information system are always getting bigger. The more their business grows the more complex their information system is. During its life, a legacy system is subjected to many maintenance activities, which cause degradation of the quality of the system and have many symptoms: pollution, embedded knowledge, poor lexicon, coupling, and layered architecture. The details of these problems are presented in [1]. All these complicate legacy systems and lower the maintainability of them. [2] Consequently, a lot of corporations or organizations choose to re-engineer the legacy system to new ones with high qualities. [4][5]

Considering that the enterprise IT architecture is important to information system maintainability, many relevant researches are carried out. Since Gartner's innovative idea in 1996, Service Oriented Architecture (SOA) and web service are becoming an integral IT landscape. More and more organizations apply SOA in their information system and their business processes are becoming more flexible and scalable. Although new development can benefit from SOA at first, it is a big challenge for legacy system maintainers to reuse their decayed wealth. Generally speaking, legacy systems are not well constructed and lack of detailed design documents. Furthermore, with years of degradation and modification, the documents are always out of date with respect of implementation and can not reflect the business logic. [6] Of course, legacy system owner can develop new SOA-based system duplicating all business logic. As we mentioned above, although the legacy systems have degenerated, they are crucial to who own them, so they normally choose to reuse and wrap the legacy codes into service components. Recent researches propose the feasible process to integrate legacy system with SOA system. [14]

Because of the complexity and dependency in business process, the relevant services have dependency and relationship as well. To achieve SOA flexibility and interoperation, it is important to distribute wrapped legacy codes and other service components into SOA IT architecture properly. Multi-level proofs its efficiency in many cases. [11] This paper addresses a multi-level architecture for organization's particular requirements, such as security, business regulations and etc. Normally, SOA is promoted by organization business side. A distribution approach which combines domain knowledge and IT analysis is researched. The problem of distributing service components into different layers is another topic in this paper. Component metrics, like Connectivity Strength (CS) and Component Complexity (CC) are used to refine components. In recent research, CS is enhanced with Weight Parameter (CSWP) to improve the precision. The paper defines its own Connectivity Strength with Weight Parameter and Frequency (CSWPF) to measure the connectivity strength for distribution. Clustering technologies, solutions of classifying data into groups, are widely used in component analysis. [3][7][8][9] An agglomerative hierarchical clustering algorithm is proposed to achieve the key step in distribution approach.

The reset of the paper is structured as follows. In section 2, we simply describe the motivation of this research. In section 3, Multi-level Service Architecture and service components criteria are given. In section 4,

0-7695-3038-9/07 $25.00 © 2007 IEEE
DOI 10.1109/ICCIT.2007.221

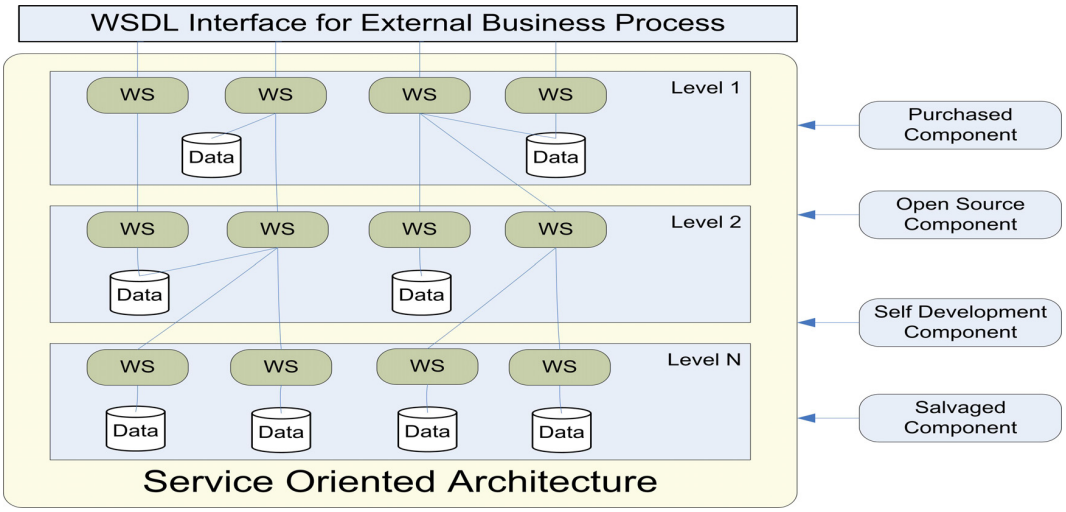


Figure 1.Multi-level Service Architecture

the approach of constructing multi-level service architecture is discussed. Section 5 describes the definition and calculation of component metric, CSWPF, which is used in clustering algorithm. Section 6 gives the clustering algorithm details. In section 7, we provide an example application for our approach. Finally, section 8 gives our conclusions and future work. Acknowledges are given to the research team and sponsors in section 9.

## 2. Motivation

It is accepted by almost all the engineers that improving maintenance is the primary goal in a re-engineering project. An organization or corporation will never plan to reengineer a well-running system, unless it is hard to maintain or the maintenance cost is high. SOA which promises better flexibility and interoperability is emerging in commercial world and adopted by many reengineering project managers. Since mid 1990s, reengineering technologies of code analysis and finding valuable business building blocks are researched well. In [15], the methods of locating and wrapping legacy codes within SOA framework are stated. Although the web services are identified and wrapped, the locations of web services should be distributed properly in the organization. In [10], Gerald Berns defined three basic metrics for measuring the maintenance process: Impact domain, Effort expanded and Second-level error rate. These metrics are also applicable in SOA IT architecture reengineering.

Security is another important concern of SOA, especially for military and financial organizations. For this reason, data and services are distributed in different layers. In [16], multi-level security for SOA is researched and applied. By doing this, most confidential data and services can be protected by high security mechanism and located in deepest level. In other case, the data in organization are maintained by different groups and the raw data may not be navigated directly by other groups, because of the regulations. Take custodian bank for example, the customers' original single trade record of a stock, like every trade's buyer, seller, price and volume, is very confidential. However, the total volume of a stock in a particular trade day is available to research groups. Although the research group is in the same organization, they should be isolated from the original data and can only access via special interfaces. It is necessary to distribute the data in different level. On the other hand, the performance of data can be much better, if the data and their related operations are put close.

The discussions above encourage us to research multi-level services architecture for enterprise IT solution. A process based on clustering technology are developed which can assist engineer to achieve multi-level from legacy applications.

## 3. Multi-level Service Architecture

In order to reengineer legacy codes and other components into a high-maintainable architecture to meet the requirements before in section 2, we propose multi-level service architecture based on SOA. It should be easy to understand, modify and enhance. We can summarize its features as follows:

- *Layered implementation.* The whole system is separated in many levels according to business logic and data security.
- *Isolated Data.* Taking the advantages of layered implementation, the data can be isolated in different

levels which have different grades and security policies.

- *Compact logic.* Since the web service access between different levels is a time-consuming task, this architecture implements relevant logic in the same level and improves the performance.

To achieve the characteristics, computer hardware multi-processors architecture practices have given good ideas. [12] Figure 1 shows multi-level service architecture. The whole architecture is based on SOA technologies and research. All service components are from sources [14]:

- *Purchased Components* that can be bought from the venders.
- *Open Source Components* that can be get from the open source communities.
- *Self Developed Components* that can be developed individually.
- *Salvaged Components* that can be extracted and taken from existing applications (legacy codes).

To large user organizations that want to apply SOA, they may use venders' service components and open source components directly to get the public information. The existing applications are reused and/or linked in SOA according to their environment dependence. The more independent the applications are, the easier to reuse them. [14] These are named salvaged components. The reengineering literature has already covered the relevant technologies. [15] For complementing the whole process, the engineers develop new components individually.

In large organizations, data normally have different privileges and maintained by different departments in different locations. There are also security requirements for some sensitive data. Distributing data in various levels according to their business and security attribute is a good solution. The levels can be applied with different control policies. In figure 1, level 1 means the web service exposed to external business process and normally has the most insensitive data. The lower level normally has higher security. The higher level can access the authorized information from the lower levels.

As we know, web service access is time-consuming. Even a single WSDL may take 500 milliseconds. To achieve the high performance, the data and relevant web services are supposed to be placed as close as possible, unless there are security or regulation issues.

Now that, an effective architecture is brought up, the problem is how to distribute data and their service components in the architecture. Applying clustering technology with connectivity strength metrics provides a possible solution. The following sections describe the distribute approach overview and detailed algorithm.

## 4. Level Distribution Approach

As all related concepts have been addressed in previous sections, this section gives the distribution approach overview.

### 4.1. Preconditions

Applying SOA in a large organization is a complex work. How to reuse legacy applications and combin them properly with services from venders, open source community and self development in multi-level SOA is a big challenge to software engineers. Our research focuses on distribution phase, and the component preparation is not included in this papers. So two preconditions are prepared as follows:

- *Well-defined service components.* As we described before, there are four sources for service components. Reuse legacy codes or new development or use a third-party service is not only IT solution but also a business integrated solution. The definition of service components should be prepared. The definition includes functionality, interface, data sources and other necessary information. The existing researches, such as software reengineering and SOA have covered them.

- *Well-defined Data Location Attributes.* Besides service components, data are another element in multi-level architecture. The attribute in this context means the information which can be used to decide the data locations. For example, in military system, the attribute can be its security policy. On the other hand, in financial system, it can be the combination within data location, security and regulation. The definitions of data attribute need the assistants from business analysts. Every related data source is investigated and defined carefully for a particular business process.

### 4.2. Approach Overview

Figure 2 shows the architecture construction process and snapshots of every stage. The step details are stated below.

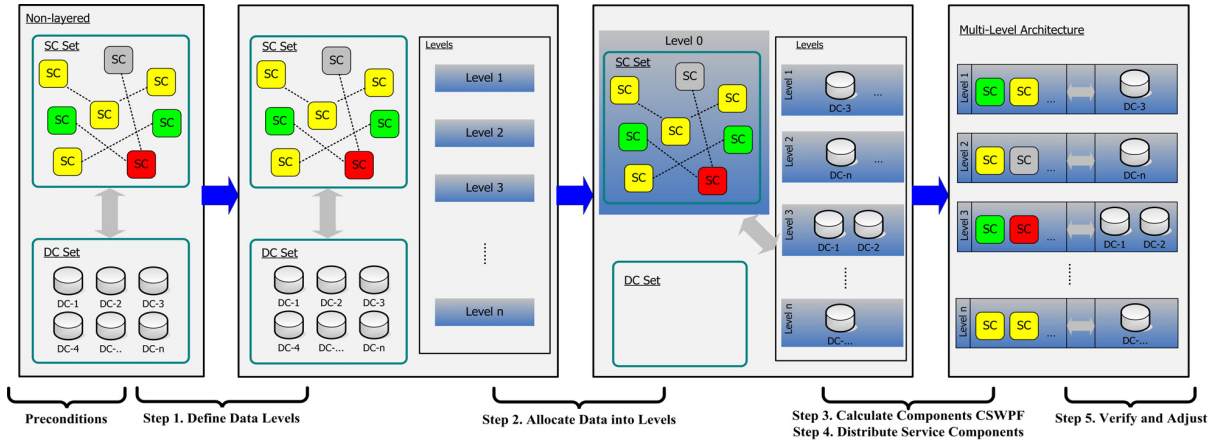


Figure 2. Multi-level Architecture Construction Process

**4.2.1. Step 1. Define Data Levels**. In Figure 1, the architecture is divided into many levels from 1 to N. It is clear that level 1 is the web services exposed to external business process, while level 2 to N are not fixed. In this step IT and domain experts should work together to complete this step according to data attributes. There are two outputs: the number of levels and their attribute.

**4.2.2. Step 2. Allocate Data into Levels.** Since the attributes of the data are prepared and the levels' attributes are defined as well, this step is just simply mapping data sources into their corresponding levels. Compared with Step 1, this step can be accomplished by IT engineers.

**4.2.3. Step 3. Calculate Components CSWPF.** CSWP is a useful component relationship metric based on CS. [3][8] By customizing and enhancing parameters of interaction parameter weights and access frequency, CSWPF can be used to measure the access relationship and efforts between service components more precise. Generally, data source can be considered a component as well. Components' CSWPF is the base of agglomerative clustering in the next step. The work in this step is the preparation and start of clustering algorithm. The definition and calculation of CSWPF are described in section 5.

**4.2.4. Step 4. Distribute Service Components.** CSWPF defined in previous step can be used to distribute the components in different levels. An agglomerative clustering is employed to distribute service components. It is a bottom-up process. The service components are distributed according to their interconnections with data sources and other related service components represented by CSWPF. The algorithm details are addressed in section 6.

**4.2.5. Step 5. Verify and Adjust.** Technically, clustering algorithm in step 4 can distribute all service components properly in multi-level SOA, because the distribution results in the optimized performance. However, the distribution may not be perfect and cause problems for some reason. According to our experience, there are three:

- *CSWPF Misestimate.* CSWPF is a metric enhanced by interaction information between components and data sources. The service parameters, returns and call frequency are based on static result and experience which may be misestimated.

- *Business Confliction.* Service component location may conflict with the current business process or tendency, although it can achieve better performance. For example, a service component, which should locates in level M for business control requirement, is distributed to level N to achieve high performance.

- *Logic Complexity.* It is not caused by distribution algorithm, but by service component construction. The construction problem can only be noticed, when the architecture is built.

In this step, IT and business experts work together to investigate, test and evaluate the built SOA system. There is no doubt that business process dominates SOA design and implementation, the verification and adjustment are necessary for the real business logic. The target of this step is making IT align business finally.

## 5. CSWPF Definition and Calculation

Software measurement has been essential to good software engineering. Many of best software developers measure characteristics of software to get some sense. For a variety of purposes, a great number of metrics are researched with the growth of software industry. Software component measurement is one of the hot

areas. [13] In [8], LEE proposed CS representing how a component is connected with other components. Extending CS with Weighted Parameter, Xinyu uses CSWP to construct components from legacy system. [3]

These practices provide good elicitations to measure components in our approach. But to SOA application, neither CS nor CSWP can be used directly. In their original applications, the components are just parts of one application. CS and CSWP mainly consider the affect of complexity to the whole system and assist software engineers to achieve high maintainability. However, in our case, service components are distributed in SOA environment. Different from components in one system, communication performance is another concern. To solve this problem, we enhance CS with access frequency to measure the connectivity strength and efforts. The definition and calculation are as following.

Assume that an SOA integration system S' consists of Service Components (SC) and Data Components (DC). Since DC can be considered a special SC, C stands for SC and DC and S' can be represented as

$$S' = \{C_1 \dots C_k\}, k \geq 1; \quad (1)$$

All components C finally belong to one of the levels (L), S' can also be represented as

$$S' = \{L_1 \dots L_i\}, i \geq 1; \quad (2)$$

CSWPF can represent the access efforts and strength between components in levels. The formulas 3 to 7 below are as follows.

$$CSWPF(S') = \sum_{L_i, L_j \in S'} CSWPF(L_i, L_j); \quad (3)$$

$$CSWPF(L_i, L_j) = \sum_{c_u \in CSET(L_i)} \sum_{c_v \in CSET(L_j)} CSWPF(c_u, c_v); \quad (4)$$

$$CSWPF(c_u, L_j) = \sum_{c_v \in CSET(L_j)} CSWPF(c_u, c_v); \quad (5)$$

$$CSWPF(c_u, c_v) = \sum_{m_x \in MSET(c_u)} \sum_{m_y \in MSET(c_v)} CSWPF(m_x, m_y) + \sum_{m_x \in MSET(c_v)} \sum_{m_y \in MSET(c_u)} CSWPF(m_x, m_y); \quad (6)$$

$$CSWPF(m_x, m_y) = \begin{cases} \left(\Pr S(m_y) + RrS(m_y)\right) * Fr(m_x, m_y), \\ \qquad if\ m_x\ calls\ m_y; \\ 0,\ otherwise \end{cases} \quad (7)$$

Where

- S', L, C and m stand for System, Level, Service Component and Service Method.
- CSET ($L_i$) means the set of components in a level $L_i$.
- MSET($c_u$) means the set of methods in a component $c_u$.
- PrS($m_y$) and RrS($m_y$) mean method y's average transfer data sizes for method parameter and return.
- Fr($m_x$, $m_y$) means the access frequency.

The units of PrS(my), RrS(my) and Fr(mx, my) are defined by software engineers according to system's characteristics. For example, PrS(my) and RrS(my) can be KB in one case or MB in other case .

## 6. Clustering Algorithm

/* Initial Preconditions*/
- $(\forall SC_i) : SC_i \in L_0$
- $(\exists! L_j) : DC_i \in CSET(L_j)$ where $1 \leq j \leq M$

/* Clustering Process*/
Iteration:
For Every ( $L_j \in L, j \geq 1$)
  Calculate $CSWPF(L_j)$;
End For

For Every $(SC_i \in L_0)$
  For Every $(L_j \in L)$
    Find $< SC_x, L_y >$ where
    $CSWPF(SC_x, L_y) = \text{Max}(CSWPF(SC_i, L_j))$;
  End For
End For

Remove $SC_x$ from $L_0$;
Distribute $SC_x$ into $L_y$;

IF $((\exists SC_i) : SC_i \in L_0)$ goto Iteration;
ELSE Exit;

Figure 3. Hierarchical Clustering Algorithm

CSWPF provides a way to measure system's potential performance. This section addresses how to utilize the metric to achieve Distribute Service Component step in the process. Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into clusters. Hierarchical Clustering and Partitional Clustering are two categories. Hierarchical algorithms find successive

clusters using previously established clusters, whereas partitional algorithms determine all clusters at once. According to application goals, agglomerative hierarchical clustering is selected.

Assume that each service component (purchased, open source, salvaged and self-development) and data in system S' is a component. In approach step 3, all data sources have been allocated to different levels according business process. The levels L can be considered groups in clustering algorithm. Data components can be used to calculate the initial CSWPF of every level L. To make the calculation much easier, we create Level 0 (L0) which is a medium and will be removed when all service components are distributed. Then service components are merged into different levels one by one. The clustering algorithm is described in Figure 3.

## 7. Application Work

This approach has been applied in a real SOA project. It attempts to reorganize the data and applications in a large bank's financial research department. In this section, we will show how the distribution approach was applied. As a research department, it needs a variety of data sources and their components. They are:

- External venders. Market data from Reuter, Bloomberg; Indicators from other research groups.
- Other Departments. Trading data from business departments.
- Internal Data source. There are some big data ware houses.

Because there are management policies in company, regulations in financial market and contracts with clients and venders, the data sources should be controlled by different rules. The relevant operations are constructed into service components. They can be provided by data venders, legacy components, and new development.

Firstly, following step 1 and 2, we divided data sources into many groups according to their control rules. The groups stand for levels. In this case, we defined 8 levels. To achieve high performance, we put the data in the same level as close as possible.

Secondly, following step 3, we defined the units for every parameter in CSWPF. The values are decided by the statistic information for a week and confirmed by domain experts. For example, method My in an indicator service component Cv has PrS(my) of 5 KB and RrS(my) of 85 KB. And method Mx in an algorithm trading component Cv accessing this method has Fr (mx, my) of 12 times/minute.

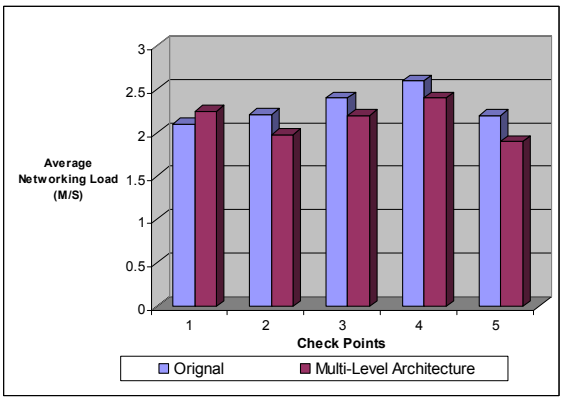


Figure 4. Networking Load Comparison

Thirdly, step 4 is applied to distribute components into our 8 levels. Since we have presented the clustering algorithm, this task is accomplished automatically.

Finally, following step 5, IT and domain experts work together to verify and adjust some components' locations. By shifting a component, engineers need to develop a new service component which can be a bridge between data source and its user components. In this step, every modification is studied case by case. The total number of service components is more than 500.

For such a complex system, it is necessary to assess our work via networking communication improvement. To be convenient, we use a simple index, average networking load, to compare the new architecture with the original one. By setting five check points in network, we get the results in figure 4. It shows that 4 spots achieve about 15% lower communication load, and only one spot is a little higher, because this spot is close to the lowest level which has many core data for almost all service components. The communication is not avoidable.

## 8. Conclusions and Future Work

Based on the real-world projects, this paper addresses the problem of integrating legacy application into SOA with special requirements, such as security, regulation and other maintenance requirements. Aiming to solve the problem, multi-level architecture is given. There are many researches on how to construct the service component. Based on those, this paper focuses on how to distribute and allocation service components to different locations to achieve high performance.

Component metric, CSWPF, is developed to evaluate the connectivity strengthen. Considering service access frequency, CSWPF is more suitable in SOA

environment. Bottom-up clustering can classify service components into different levels one by one by achieving maximum CSWPF for every level and minimum CSWPF between levels in the whole architecture. The innovative distribution approach with metric and clustering algorithm has been successfully applied to real project and proofs its effective.

Our future work will focus on enhancing CSWPF calculation parameters evaluation and refining the approach, the CSWPF calculation parameters, such as PrS(my) and RrS(my), are gained from engineers' experience and statistics on the system. The different parameter estimation may result in different distribution. Although the estimations are relatively correct now, more researches will be carried out to get a sophisticated methodology which can guide software engineer to estimate more precisely. The last approach step, verify and adjust, is accomplished by manually, now. Developing a framework and relevant tool to verify the clustering output is another research direction for us.

## 9. Acknowledgements

This research is collaborative efforts between Zhejiang University, China and State Street Corporation, USA who is one of the top financial services providers world-wide and custody 15 percent of the world's tradable assets. We are thankful for the help from State Street Corporation in our research. The contents of this paper are the opinions and conclusions of the authors only and do not necessarily represent the position of State Street Corporation or its subsidiaries, affiliates, officers, directors or employees.

## References

[1] A. Bianchi, D. Caivano, V. Marengo, and G. Visaggio: "Iterative Reengineering of Legacy Systems", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 3, MARCH 2003

[2] N. F. Schneidewind, C. Ebert: "Preserve or Redesign Legacy Systems?", IEEE SOFTWARE July/August 1998

[3] Xinyu Wang, Jianling Sun, Xiaohu Yang, Chao Huang, Zhijun He, Srinivasa R. Maddineni,: "Reengineering standalone C++ legacy systems into the J2EE partition distributed environment", International Conference on Software Engineering 2006, ICSE 2006, Proceeding of the 28th international conference on Software engineering. pp: 525 - 533

[4] G. Snelting and F. Tip: "Reengineering Class Hierarchies Using Concept Analysis", In ACM Trans. Programming Languages and Systems, 1998, pp: 15-20.

[5] HARRY. M. SNEED: " Planning the Reengineering Legacy Systems", IEEE SOFTWARE, July, 1995

[6] G. Murphy, D. Notkin, K. Sullivan: "Software Reflexion Models: Bridging the Gap between Source and High-Level Models", Proceedings of the ACM SIGSOFT '95, Washington, D.C., 1995

[7] Jain, H.; Chalimeda, N.; Ivaturi, N.; Reddy, B: "Business component identification - a formal approach", Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International 4-7 Sept. 2001 pp.183 - 187

[8] Eunjoo Lee Byungjeong Lee Woochang Shin Chisu Wu: "A reengineering process for migrating from an object-oriented legacy system to a component-based system", Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International 3-6 Nov. 2003 pp. 336 – 341

[9] Eunjoo LEE, Woochang SHIN, Byungjeong LEE, and Chisu WU: "Extracting Components from Object-Oriented System: A Transformational Approach", IEICE Transactions on Information and Systems Number 6, 2005. Volume E88-D, pp. 1178-1190

[10] G. Berns, "Assessing Software Maintainability", Comm. ACM, Jan. 1984, pp. 32-49.

[11] Merijn de Jonge: "Build-Level Components", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 7, JULY 2005, pp: 588 -600

[12] David A. Patterson, John L. Hennessy: "Computer Architecture A Quantitative Approach", China Machine Press, China, 1999

[13] Norman E.Fenton, Shari Lawrence Pfleeger: "Software Metrics: A Rigorous & Practical Approach", Thomson Press, 1997.

[14] Harry M. Sneed, "Integrating legacy Software into a Service oriented Architecture", Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06) Volume 00, 22-24 March 2006 pp:11

[15] Bodhuin, T/Guardabascio, E./ Totorella, M.: "Migrating COBOL Systems to the WEB" , Proc. Of 9th WCRE-2002, IEEE Computer Society, Richmond Va., Nov. 2002, p. 329

[16] Ramasamy, H.V.; Schunter, M.: "Multi-Level Security for Service-Oriented Architectures", Military Communications Conference, 2006 (MILCOM 2006)