# Transforming Legacy Energy Management System (EMS) Modules into Reusable Components: A Case Study

Farrokh-Fattahi*,Ali-Zeraatparvar*,Naser-Tabatabei*,Arif-Hashimov**
*Azarbaijan Higher Education and Research Complex (AHERC)-Tabriz,Iran
**Azarbaijan national academy of sciences
E-mail:Fattahi1340@yahoo.com

***Abstract:*** A two-layer adaptive wrapping technique is proposed in this paper to migrate legacy Energy Management System (EMS) modules into reusable components, and therefore transforming the legacy system into a component-based system. Meanwhile, an XML based database wrapping technique is proposed to provide a flexible data exchange solution. These techniques help not only maintaining the reliability but also increasing the reusability of the legacy EMS during the software evolving process.

***Keywords:*** Distributed Component Software Architecture, Architecture Evolution, Wrapper, Energy Management System (EMS).

## 1. MOTIVATION

Legacy software systems exist everywhere. In some industrial domains, such as power and oil & gas industry, the software systems are very large and complex, and have proved to be highly reliable during their long time service. None or little re-engineering has been conducted on these systems. They are considered as legacy systems.

Currently, there is an increasing need to reengineer the legacy systems due to the changes in the industries and the markets. For example, the power industry is undergoing a dramatic restructuring, which is evolving from vertically integrated utilities into deregulated power markets. As a consequence, existing legacy system, particular the Energy Management System (EMS) needs to be reengineered to meet the needs of the market and customer: old functionality of the system needs to be updated, and new features are desired. Some of the most important new requirements include the support of electronic business to provide web access and good information supporting system. Also since PCs with windows operating system have gained popularity, multi-platform support becomes necessary.

On the other hand, the disciplines of software engineering and software architecture have made tremendous progress in the past decade. Software and software tools have evolved mainly along five dimensions: the expressive power of program languages, modularity, platform independence, reusability, and verification.

This paper focuses on efficiently migrating legacy systems using contemporary software engineering and software architecture technology to meet the current needs of the market and customers. Large amount of research has been published on this subject. Software tools have been done to support the transformation of legacy systems as well. Generally there are two types of approaches: the white-box solution and the black-box solution. Weiderman [1] categorized the white-box and the black-box solutions as the following: “the white-box transformation encompasses a form of reverse engineering that emphasizes deep understanding of individual modules and internal restructuring activities. The black-box transformation encompasses a form of reverse engineering that emphasizes shallow understanding of module interfaces and wrapping activities”. The wrapping technique with the potential to couple with distributed component technology, has gained much support and has large applications in some domains. Different wrapping techniques dealing with different level modification of legacy systems have been proposed in Sneed [2]. In this paper, the usage and benefits of wrapping techniques are further explored. A two-layer adaptive wrapping technique is proposed to increase the reusability of the modules in the legacy EMS and reduce the maintenance as well as development work. Meanwhile, an XML based database wrapping technique is proposed to provide a flexible data exchange solution.

## 2. SOLUTION STRATEGY

In the deregulated power industry, the power market structures could be different. Generally, two alternatives (central model and de-central model) are widely used as shown in Fig. 1 and Fig. 2.

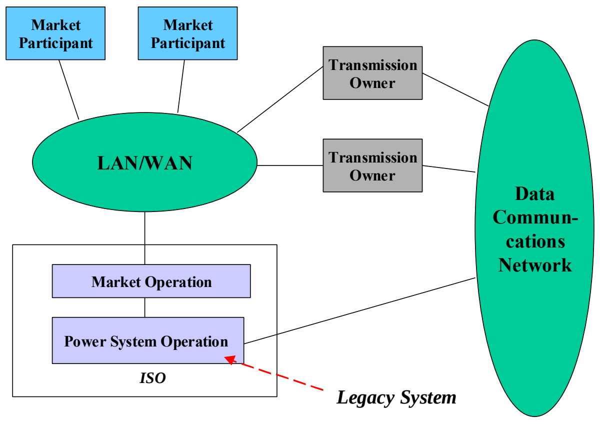


Fig.1. Market Structure Alternative 1 (Central Model)

In the central model, the Independent System Operator (ISO) is responsible for both the market operation and the power system operation. Whereas in the de-central model, the ISO is only responsible for the power system operation, and the market operation is performed by Power Exchange (PX) and Schedule Coordinator (SC). The legacy systems in the central and the de-central models are identified in Fig.1 and Fig. 2.

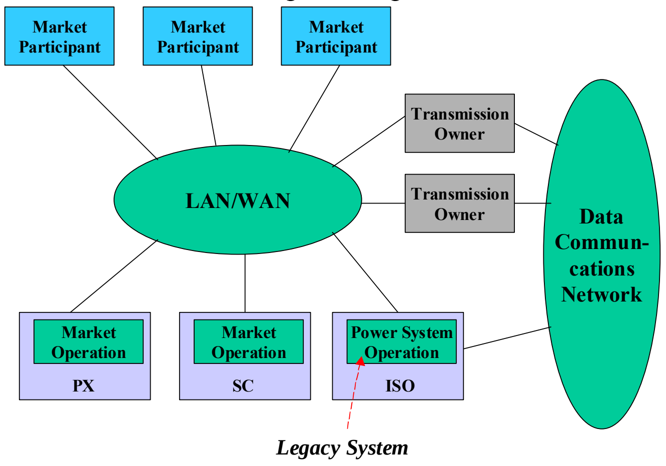


Fig. 2. Market Structure Alternative 2 (De-central Model)

EMS is a large, complex, and reliable real time software system. Therefore the following strategies are considered to maintain the reliability and to increase the reusability of the system when designing a time-to-market system evolution solution.

**STRATEGY 1: Least possible changes to the existing system is desired.** Because EMS is large and complex, it is impossible for the vendors to build a new system from scratch, and substantial changes in the software system may need a large amount of testing work, which may not be feasible due to the company's policy and budget limitations.

**STRATEGY 2: Errors of newly developed component should not transgress into the legacy system and affect the reliability of the whole system.** High reliability is one of the most important requirements for EMS. Any fault of the software system during its service may cause disasters. The functionality and reliability of the legacy system should be preserved.

**STRATEGY 3: It should be a cost-effective solution.**

The above strategies guide to design a new system to satisfy the requirements from the market and the customers. The new system contains two parts. One part is the existing EMS, and the other is the Market Operation System (MOS). The EMS, running on UNIX, monitors and controls the real-time operation of power system. The MOS can run on multiple platforms, such as UNIX or MS-Window. An example of the physical diagram of the new system is shown in Fig. 3.

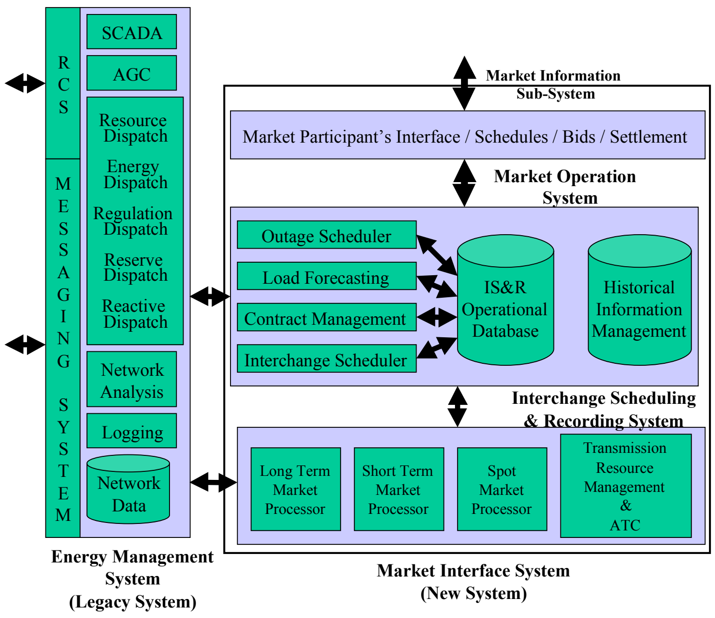

RCS
MESSAGING SYSTEM
SCADA
AGC
Resource Dispatch
Energy Dispatch
Regulation Dispatch
Reserve Dispatch
Reactive Dispatch
Network Analysis
Logging
Network Data
Energy Management System (Legacy System)
Market Information Sub-System
Market Participant's Interface / Schedules / Bids / Settlement
Market Operation System
Outage Scheduler
Load Forecasting
Contract Management
Interchange Scheduler
IS&R Operational Database
Historical Information Management
Interchange Scheduling & Recording System
Long Term Market Processor
Short Term Market Processor
Spot Market Processor
Transmission Resource Management & ATC
Market Interface System (New System)

Fig. 3. Physical Diagram of MOS and EMS

This design has the following advantages:

- It satisfies the requirements of the power industry restructuring.
- It enables the EMS vendors to preserve the old EMS, in other words, the company assets.
- It enables the MOS vendors to provide a flexible software system for different power market structures.

## 3. TWO-LAYER ADAPTIVE WRAPPING TECHNIQUE

### 3.1. Overview of Legacy System

To clearly understand the problem and our approach, it is important for us to do some studies on the legacy system and to understand the architecture of the legacy system.

A typical EMS design uses a domain-specific approach. The system may contain the following domains: Security Analysis, Energy Schedule and Accounting, Automatic Generation Control, System Control And Data Acquisition (SCADA), etc. Each domain may contain many subsystems as well. For example, the Security Analysis domain may have the following subsystems: State Estimator, Dispatchers' Load Flow, Contingency Analysis, etc. Each of the subsystem is designed as an independent process with a unique task id on the UNIX platform. The data exchange only happens between the subsystem and the database. The subsystems can be scheduled by the other subsystems, the real-time core program or by the operator through the display. When a subsystem is scheduled, it gets the parameters sent by its originator from the shared memory. The parameters contain the following information: task-id, database no., database mode no., which tells the subsystem the data source.

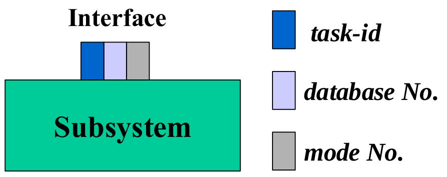


Fig.4. Interface of Subsystem

In an EMS, the software system consists of layers of functions: the Basic System Functions (BSF), the Application Supporting Functions (ASF) and the Application Functions (APF). The BSFs contain many small and simple subroutines. It is time-consuming job to understand those subroutines to be redesigned into a white-box solution. The ASFs contain usually large and useful subroutines, such as complex mathematical algorithms. It is beneficial to reuse these functions, but it is time-consuming and labor-intensive to understand the codes because the functions are heavily nested with the BSFs. The APFs are designed as independent executable tasks with unique task id and uniform interface.

## 3.2. A Two-Layer Adaptive Wrapping Technique

A white-box solution can increase the reusability of legacy systems with efforts to understand the modules at code level. On the other hand, although the black-box solution may require fewer efforts than the white-box solution, it may not be good in terms of increasing the reusability of legacy systems. A solution, which combines the best of both solutions, is the most preferred.

A two-layer adaptive wrapping technique is illustrated in Fig. 5, which uses a transformation technology to combine a white-box wrapping and an adaptive black-box wrapping techniques.

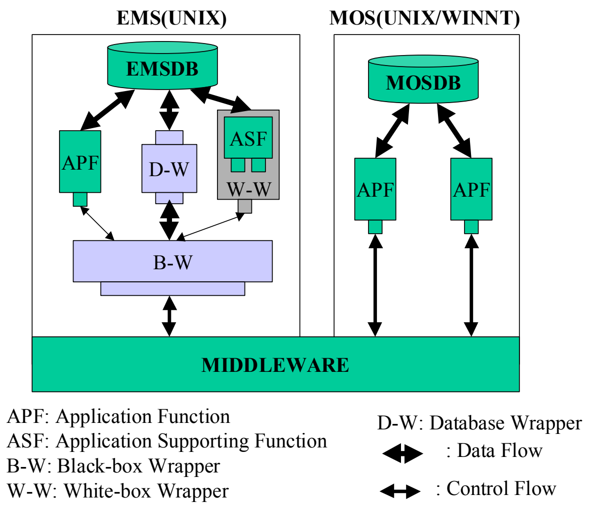


Fig.5. A Two-Layer Adaptive Wrapping Technique

In layer 1, a white-box wrapper is developed for each of the ASFs, which is a main program to make the ASF an independent function with uniform interface as that of the APFs. Then, the wrapped ASFs are compiled to become executable tasks with newly assigned task id. In layer 2, an adaptive black-box wrapper is used to transform the Wrapped ASFs (WASF) and the APFs into reusable components.

(1) Algorithm for the white-box wrapper

The ASFs are subroutines for complicated mathematical algorithms, which are very useful but difficult to implement and test. The ASFs are called by the APFs. The interfaces of the ASFs are variables exchanged through the shared memory or through the calling arguments. For example, the ASF for Linear Equation Solver is to solve the large linear equation ([A]*[x] = [b]) by using sparse matrix technique. In some of the implementation, six arrays (four arrays for sparse storage of [A], one for [x] and one for [b]) are exchanged as the calling arguments between calling functions and the equation solver ASF. The ASF was implemented in FORTRAN, and is used by several subsystems in EMS and will continue to be used by MOS functions.

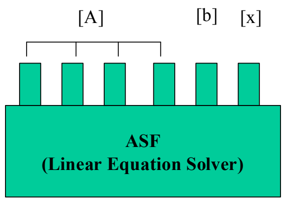


Fig. 6. Interface for ASF (linear equation solver)

The white-box wrapper facilitates the ASF with the same interface as that of the APFs. Each ASF needs a white-box wrapper. Together with the white-box wrapper, ASF is compiled into an executable process with a newly assigned task-id and the interface as that shown in Fig.4. The Wrapped ASF (WASF) exchanges data only with EMS Database (EMSDB).

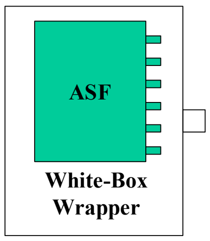


Fig. 7. White-box Wrapping Technique

In the white-box wrapper, the data needed for the ASF are obtained from EMSDB and decomposed/composed into calling arguments for the ASF. After calling the ASF, the result of the ASF is stored back to the EMSDB. The algorithm for the white-box wrapper is:

```
White-Box-Wrapper-ASFx
{
      get the parameters sent by the
            calling function from the
            shared memory;
            identify the data source and get
                  data from EMSDB;
            decompose/compose local interface
                  of ASFx;
            call ASFx;
            get result of ASFx;
            store the result into EMSDB;
}
```

(2) Algorithm for the adaptive black-box wrapper

The wrapped ASFs and the APFs are executable processes with unique task id and uniform interface. An adaptive black-box wrapper is used to transform them into reusable components.

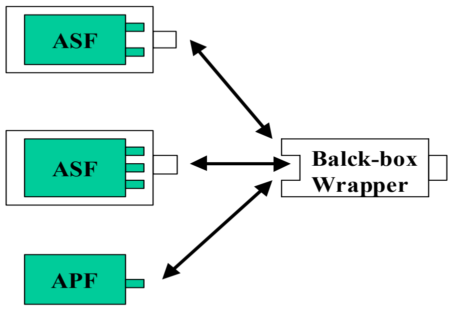


Fig. 8. Black-box Wrapping Technique

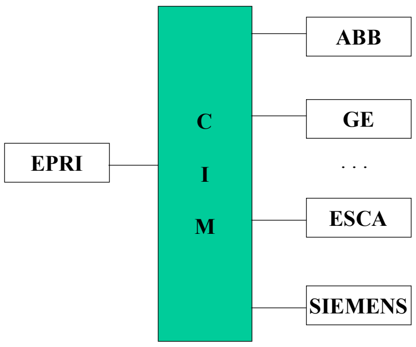
In the adaptive black-box wrapper, the task-id of the WASF or the APF is passed as an argument from the MOS Application Function (MOSAPF), which is used to schedule the specified WASF or the APF. Therefore, only one black-box wrapper is needed for all the WASFs and the APFs, which reduces development and maintenance works. For example, when a WASF or APF is developed or deleted, no much work is needed for the adaptive black-box wrapper. So, this black-box wrapper is independent of the modules in the legacy system, and is adaptive to modifications in the legacy system. The algorithm for the white-box wrapper is:

```
Black-Box-Wrapper(WASFx/APFx_id, data
from MOSAPF, data to MOSAPF)
{
      get request from MOSAPF;
      translate input of MOSAPF into
            EMSDB;
      prepare the parameters for the
            WASFx/APFx in the shared
            memory;
      schedule WASFx/APFx;
      translate result in EMSDB into
            MOSDB;
      return to MOSAPF;
}
```

## 4. A DATABASE WRAPPING TECHNIQUE FOR DATA EXCHANGE PROBLEM

Through the two-layer wrapping technique, the modules (including APFs and ASFs) become reusable components to the MOS. However, the database modeling in different vendors' EMS could be different. Since, different MOS vendors would develop a system that could be easily integrated with different EMS, a universal applicable data exchange solution will be very important.

Recently, the data exchange problem has been paid more attention in power industry. Different EMS vendors have different database models, however, some advanced power system analysis functions can not be supplied by the EMS vendors, which will be developed by other companies or research institutes, such as Electric Power Research Institute (EPRI). US EPRI has faced that difficulty before, and they have proposed to standardize the database model in power industry. Some industry standard is under establishment. The US EPRI has also developed a CIM



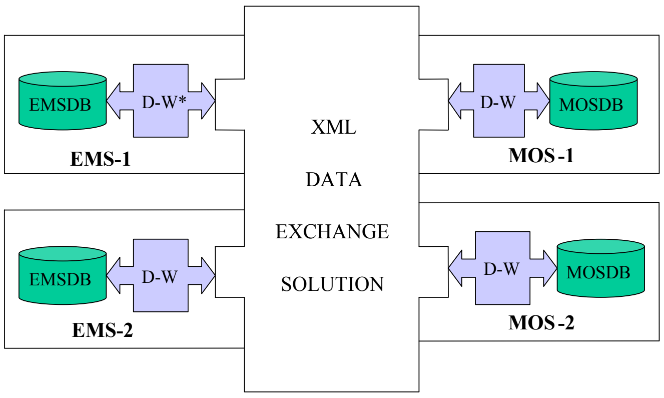
system to provide an interface tool to connect different EMS vendors' products in 1999.

Fig. 9. EPRI's CIM Solution

However, eXtensible Markup Language (XML) has received a support from the whole software industry for its simplicity of syntax and low cost tools. It has been recognized as the most flexible data exchange solution for heterogeneous database. Here, we propose to use XML and the EPRI's database definition to support the data exchange between EMS and MOS.

In this solution, the power industry standard on database modeling is used as the global Document Type Definition (DTD), and the data transfer between EMS and MOS are XML data file with the structure as defined in DTD. In EMS and MOS, Database Wrappers are developed in EMS and MOS to transforms EMS/MOS database tables to/from XML data files. The Database Wrapper algorithm is:

```
Database-Wrapper(XML data file, database
no, database model, WASFx/APFx id,
Database-to-file flag)
{
      identify the right database tables
            based on the information of
            database no, database
            model, WASFx/APFx id;
        if Database-to-file flag is true
            convert the XML data file
            to the data in the database
            tables based on  the global
            DTD;
        else
            convert the data in the
            database tables into XML
            data file based on  the
            global DTD;
        end if
}
```



* D-W: Database Wrapper

Fig. 10. XML-based Data Exchange Solution

## 5. ARCHITECTURE ALTERNATIVES

### 5.1. Remote Call without Using Middle Ware

To demonstrate the proposed two-layer adaptive wrapping technique, we did a very simple example on remote call from MS-WIN95 platform on DELL machine to UNIX platform on Sun Sparc20 machine by using Remote Procedure Call (RPC) in ANSI C language. The Sun Sparc20 machine acted as the RPC server and the WIN95 machine as a client. Sun Solaris has the OSF RPC installed. The interface program was compiled by using command “rpcgen”, and the remote function program and the skeleton were compiled and executed on the Sparc20 machine. Our first test is to compile the stub program and the main program by using MSRPC on the MS-WIN95 machine. The first test failed. Then we did a second test by replacing MSRPC by OSF RPC library in the WIN95 machine. This time, the remote call is successful, which indicates the proposed scheme is feasible with the same RPC protocol.

From this test, we observed that the proposed two-layer adaptive wrapping technique is feasible. However, we also learnt that the compatibility of protocol is very important from our first failure.

### 5.2. Recommendation on Selection of Middle Ware Alternatives

Through the two-layer wrapping techniques, the EMS modules become reusable components, which could be located in different machine as that of the MOS. The distributed component technology supports this software architecture. Recently, the three most popular distributed component architecture specifications are: OMG's Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM), and JavaSoft's Java/Remote Method Invocation (Java/RMI). A brief comparison of these three specifications is in Table 1.

Table 1. COMPARISON OF CORBA, DCOM and JAVA/RMI

| | **Supporting of Multi-Platform** | **Supporting of Multi-Language** |
|---|---|---|
| **CORBA** | Yes | Yes |
| **DCOM** | Good for MS-Windows | Yes |
| **Java/RMI** | Yes | Only Java |

On one hand, because the EMS runs on UNIX and the MOS could run on UNIX or MS-Windows, the candidate middle ware is required to support multi-platform. On the other hand, the performance of Java language is not good, and the C++/C and FORTRAN is still main language used in EMS and MOS even though Java may be used to develop the web applications in MOS, so, the candidate middle ware is required to support multi-language. Because of these two requirements, only CORBA is the suitable middle ware. Here, we recommend CORBA as the middle ware used between EMS and MOS.

## 6. CONCLUSIONS

We proposed a two-layer adaptive wrapping technique and a database wrapping technique to transform the legacy EMS modules to reusable components, which increases the reusability and preserves the reliability of legacy software system. In layer 1, a white-box wrapper is used to provide an uniformed interface for the ASFs and make them executable processes. In layer 2, a adaptive black-box wrapper is used to transform the WASFs and the APFs into reusable components. The proposed two-layer adaptive wrapping technique provides a good cost-effective solution for legacy system transformation, and it has been proved to be a feasible solution through our testing.

The following future work is needed:

- Extension of this two-layer adaptive wrapping technique into other domains.
- Universal definition of EMS components;
- Design of a software architecture for MOS, which is universally applicable or easily extendable to different power market models.
- A solution for EMS data exchange: a case study.

- Data-centric legacy EMS migration to web-based system: a case study.
- Transforming legacy EMS into web-enabled system: a case study.
- An architecture for web-based EMS.

## 7. ACKNOWLEDGE

The authors gratefully acknowledge the support from Texas Energy Resource Program at Texas A&M University and EPRI/ARMY CONSORTIUM.

The authors also gratefully acknowledge the support from NASA Independent Validation and Verification (IV&V) Facility and Jet Propulsion Lab. (JPL) to develop XML-based database wrappers for data exchange.

## REFERENCE

[1]. Nelson Weiderman, etc, "Implications of Distributed Object Technology for Reengineering," Technical Report, CMU/SEI-97-TR-005, ESC-TR-97-05.

[2]. Harry M. Sneed, "Encapsulating Legacy Software for Use in Client/Server Systems," IEEE Proceedings of WCRE'96, 1996.

[3]. K.H. Bennett, etc., "Decision model for legacy systems," IEE Proc.-Softw., Vol. 146, No. 3, June 1999.

[4]. M.Calder, etc., "Hybrid approach to software intwerworking problems: managing interactions between legacy and evolving telecommunications software," IEE Proc.-Softw., Vol. 146, No. 3, June 1999.

[5]. Norman F. Schneidewind, etc., "Preserve or Redesign Legacy Systems?" IEEE Software, July/August 1998.

[6]. G. T. Heineman, A. Mehta, "Architecture Evolution of Legacy Systems," COMPSAC'99, Phoenix, USA.

[7]. R. C. Dugan, T. E. McDermott, "Design of Interfaces for Power Systems Analysis Components," IEEE SM'99, Edmonton, Canada.

[8]. S. Talukdar, "The Next Software Revolution," IEEE SM'99, Edmonton, Canada.

[9]. P. Hirsch, S. Lee, "Security Application and Architecture for an Open Market," IEEE Computer Applications in Power, Vol. 12, No. 3, July 1999, pp26-31.
[10]. Gopalan Suresh Raj, "A Detailed Comparison of CORBA, DCOM and Java/RMI," http://www.execpc.com/~gopalan/misc/compare.html.

[11]. Bob Ducharme, XML – The Annotated Specification, Prentice Hall PTR, New Jersery, 1999.

## ENERJİNİN İDARƏ ETMƏ SİSTEMİNİN ÇOXSAYLI İSTİFADƏ KOMPONENTLƏRİNƏ ÇEVRİLMƏSİ. HAZIRLANMIŞ NÜMUNƏNİN ŞƏRHİ

**Fəttahi F., Zəraatpərvər A., Təbatabai N., Həşimov A.**

Məqalədə enerjinin idarə olunması sisteminin çoxsaylı istifadə təşkiledicilərinə çevrilməsi texnikası təklif olunur. Təklif olunan texnika verilənlərin mübadiləsi problemlərinin həllini təmin edir. Bu üsullar etibarlığı təmin etməklə yanaşı eyni zamanda proqramm təminatının inkişafı prosesində idarə etmə sisteminin çoxsaylı istifadə olunma imkanlarını artırır.

## ПРЕОБРАЗОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ЭНЕРГИЕЙ В КОМПОНЕНТЫ МНОГОКРАТНОГО ИСПОЛЬЗОВАНИЯ. ПРИМЕР КОНКРЕТНОГО ИСПОЛНЕНИЯ

**Фаттахи Ф., Зераатпарвар А., Табатабаеи Н., Гашимов А.**

В статье предложена адаптивная техника двухслойного свертывания для перевода Системы Управления Энергией в компоненты многократного использования. Предложенная техника обеспечивает гибкое решение проблемы обмена данными. Эти методы позволяют не только поддерживать надежность, но также и увеличивать возможности многократного использования системы управления в течение процесса развития программного обеспечения.