Journal of The Korea Society of Computer and Information
Vol. 21 No. 11, pp. 93-101, November 2016

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2016.21.11.093

Master Data Management API for Assembled ERP Tools

Jin-Kyung Park*, Mi-Su Gim**

Abstract

In this paper, we propose a new model, assembled ERP, to analyze the reason why the small and medium sized companies have a low ERP implementation rate and to solve this. As a result of our analysis, the high initial costs and time required for implementing ERP keep such companies from implementing it as they have relatively limited resources. The assembled ERP model, however, reduces the scale of ERP by developing the ERP for the unique business of each and connecting a few 3rd-Party applications as a module to make ERP for common work. Also it enables easier data utilization by drawing up an API list that connects the 3rd-Party applications to the developed ERP when the user lacks knowledge in the master data of the 3rd-Party applications. Therefore it lets the small and medium sized companies introduce the ERP more easily by reducing the costs for ERP implementation and user training.

▸ Keyword : ERP(Enterprise Resource Planning), EAI(Enterprise Application Integration), API(Application Programming Interface), Master Data, Transaction Data, DB Management, Data Management, Assembled ERP

I. Introduction

90년대 후반 이후 정보통신의 발달에 따라 기업은 정보 기술을 활용하여 자신들이 가진 자원을 효율적으로 관리하기 위해 노력하였으며, 이를 위해 ERP(Enterprise Resource Planning) 시스템을 도입하는 기업이 증가했다. ERP를 도입한 기업은 업무 처리 및 재고 관리 효율이 높아지고, 1인당 매출액이 증가하는 등 그 효과를 보고 있다.

이처럼 ERP의 효과가 입증되었음에도 불구하고 2014년 중소기업의 ERP 도입 비율은 38.4%로 절반에도 미치지 못하고 있는데, 중소기업이 ERP 도입을 쉽게 결정하지 못하는 결정적인 이유는 ERP 도입에 필요한 비용이 커서 중소기업이 감당하기에는 부담스럽기 때문이다. 현재 ERP 시스템을 도입할 때 완성된 ERP 패키지를 구입한 후 기업의 업무 상황에 맞도록 커스터마이징 과정을 거친 다음 그 사용법을 직원에게 교육하여 업무에 투입하는 방식을 사용하고 있다. 이 때, 커스터마이징과 직원 교육에 소모되는 비용이 전체 ERP 도입 비용의 절반 가까이를 차지하고 있다.[1][11]

따라서 이러한 비용을 줄이기 위해 기업에서 ERP 시스템을 직접 개발하여 사용하는 조립형 ERP 모델을 대안으로 제시할 수 있다. 조립형 ERP 모델은 직원 교육과 커스터마이징에 들어가는 비용을 최소화 할 수 있으며, 개발단계에서부터 기업의 업무 체계에 맞추어 진행하기 때문에 방대한 기능으로 인한 혼란을 줄일 수 있다는 장점이 있다. 하지만 조립형 ERP 모델을 도입할 경우 업무용 데이터베이스를 재구성해야하기 때문에 개발기간이 늘어날 수 있고, 때문에 개발 비용이 증가하는 문제가 발생할 수 있다.

이러한 문제점을 해소하기 위하여 본 논문에서는 Opensource ERP Tool인 Adempiere에 정의된 데이터를 활용하여 특정 기업 프로세스에 필요한 Transaction Data와 Master Data를 설정하고, 사용자가 Master Data의 내역을 정확히 알지 못해도 쉽게 데이터를 관리할 수 있는 API를 정의하고자 한다.

• First Author: Jin-Kyung Park, Corresponding Author: Mi-Su Gim
*Jin-Kyung Park (willowleaf@uos.ac.kr), Dept. of Electronic&Electrical Computer Engineering, Univ. Of Seoul, Korea
**Mi-Su Gim (miso4u@uos.ac.kr), Dept. of Electronic & Electrical Computer Engineering, Univ. Of Seoul, Korea
• Received: 2016. 08. 25, Revised: 2016. 09. 06, Accepted: 2016. 11. 18.

II. Background

관련 연구에서는 ERP의 정의와 역사, 그리고 ERP 시스템 구축시의 장점 등을 다루고 있다. 또한, EAI 방식의 정의와 그 구성요소, 그리고 API의 정의와 그 기능에 대해서도 다루고 있다.

1. ERP

ERP는 Enterprise Resource Planning(전사적 자원관리)의 약자로, 1970년대에 제조 기업에서 사용된 MRP(Material Requirement Planning : 자재 소요 관리)를 현대 산업에 맞게 발전시킨 통합 자원 관리 시스템이다.[2]

이후 MRP II(Material Resource Planning : 제조 자원 관리)를 거쳐 1990년대 이후에 ERP가 보급되기 시작했으며, 정보 기술이 발전한 2000년대 이후에는 다양한 툴을 활용한 통합 정보관리 시스템으로 발전했다.

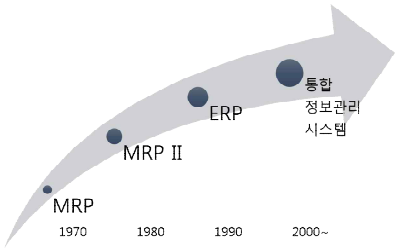

Fig. 1. development of ERP[3]

2001년, 산업자원부에서 ERP 시스템을 도입한 120여개의 국내 중소기업을 대상으로 ERP의 효과에 대한 설문조사를 진행했다. 조사 결과에 따르면, ERP 시스템 정착 이후 고객납기 응답기간, 재고보육기간, 월차마감기간은 감소하였고 직원 1인당 매출액은 증가하여 기업에 긍정적인 효과를 가져왔음을 볼 수 있다.[1]

기술정보진흥원도 2002년부터 2006년까지 생산정보화지원사업에 참여한 기업들을 대상으로 ERP 도입의 효과에 대한 설문조사를 진행했는데, 이 조사 결과에 따르면 데이터 도입시간이 88.3%, 서류 작업시간은 80.0%, 작업준비 시간은 39.6% 감소했으며 전체 매출액은 평균 18.9%, 직원 1인당 노동생산성은 15.9% 증가하는 등 ERP 도입이 기업 운영에 도움이 된다는 것을 알 수 있다.[4]

그럼에도 불구하고, 2014년을 기준으로 한 국내 중소기업의 ERP 도입 비율은 약 38.4%에 그친다. 미래창조과학부에서 2001년부터 중소기업의 ERP 도입을 위해 지원을 아끼지 않았던 것에 비하면 증가율이 저조하다. 많은 중소기업들은 도입을 망설이는 이유로 예산 부족과 업무 환경에 맞지 않아 위험부담이 있음을 꼽았다.[1][5][12]

2. EAI

EAI는 Enterprise Application Integration(기업 애플리케이션 통합)의 약자로, 기업 내부의 애플리케이션 통합과 데이터 동기화에서부터 기업과 기업이 원활한 업무 진행을 위해 애플리케이션을 통합하는 것을 의미한다.[6]

EAI는 기능적인 면에서 미들웨어와 유사한 점이 있다. 하지만 미들웨어가 애플리케이션간의 연결을 Point to Point 방식으로 구성하여 독립적으로 관리하는 것에 반해 EAI는 하나의 중심 애플리케이션을 두고 사용자의 필요에 따라 여러 애플리케이션을 Hub & Spoke 방식으로 연결하여 통합적으로 관리한다는 차이점이 있다.[7]

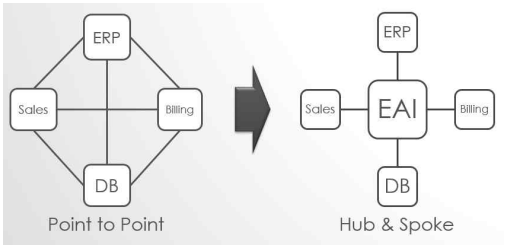

Fig. 2. Point to Point, Hub & Spoke Method[3]

EAI의 기본 구성 요소로는 Adapter, Message Queue, Message Distribution, Data Broker, Process Integration 등을 들 수 있으며 각각의 구성요소에 대한 설명은 Table 1과 같다.

Table 1. Components of EAI[8]

Component	Explanation
Adapter	This module is developed and is offered to connect directly with existing ERP and Applications. Developers can further develop adapters at their needs.
Message Queue	Located between the adapter and broker, this performs asynchronous interactions between different applications.
Message Distribution	This distributes messages through public standards such as the Internet and intranet.
Data Broker	This integrates various types of data formats that different applications create.
Process Integration	This is a software module that provides automated business logic to enable data to be applied to real-world tasks. Graphics tools are provided for designing and performing business processes.

3. API

API는 Application Programming Interface의 약자로, 서로 다른 소프트웨어간의 자료 교환 및 통신을 위한 인터페이스로 사용하기 위한 코드이다. 통신을 위한 복잡한 기능을 함수 호출로 간단히 사용할 수 있도록 구성되어 있다.[9]

API를 사용하면 각 애플리케이션의 개발환경과는 별도로 사용 중인 운영체제에 친화적인 개발을 할 수 있으며, 서로 다른 포맷을 사용하는 애플리케이션간의 데이터 교환을 쉽게 할 수 있다.

개발자는 API의 헤더와 파라미터, 기능명세를 참고하여 애플리케이션 개발 시 필요한 기능을 사용할 수 있다. 애플리케이션 사용자가 동작을 행하면 해당 동작에 대응하는 API가 호출되고, API는 정상적인 호출인지를 검사한 후 기능을 수행하여 결과를 반환한다.

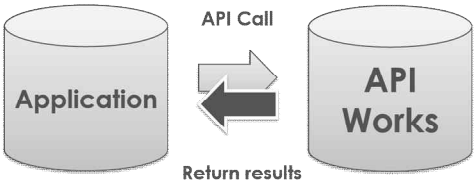

Fig. 3. API works[3]

III. Master Data Management API

Opensource ERP Tool인 Adempiere에 존재하는 여러 가지 업무 프로세스 중 하나인 Sales Order Process와 연관된 Data Table을 참조하여 필요한 Table을 선별한다. 이중 다양한 업무에서 공통적으로 사용할 수 있는 Table을 지정하여 Master Data로 분류한다.

헤더와 파라미터로 이루어진 API를 정의하고 그 List를 개발자에게 제공한다. 개발자는 API List를 참고하여 필요한 API를 호출, 데이터베이스에 간접적으로 접근할 수 있다. 개발자는 Master Data의 내부에 직접 접근할 수 없으며, 오직 API를 통해 값을 Select하거나 Update할 수 있다. 직접 접근을 제한함으로써 사용자가 임의로 Table을 추가하거나 수정할 수 없도록 하여 Process 진행에 필요하지 않거나 중복되는 Data를 가지는 Table이 생성될 가능성을 줄일 수 있다.

1. Data Model

1.1 Adempiere

본 논문에서 사용하는 Data Table은 모두 Adempiere에서 정의된 Table을 참고하였으며 이 중 Sales Order Process에 속하는 Table을 데이터 모델로 특정하고 연구를 진행하였다. 따라서 이후에 정의하는 API는 Adempiere의 User Interface 상에서 동작하는 것을 가정하고 설계하였다.

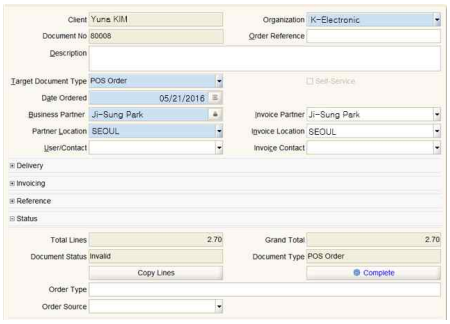

Fig. 4. Adempirere User Interface

1.2 Transaction Data

Transaction Data는 주문, 판매, 입금, 출금 등 외부 거래가 발생했을 때 그에 대한 변경 내용을 기록하는 데이터이다. 발생 데이터 혹은 변경 데이터라고도 부른다. 주로 데이터 변경이 잦은 테이블이 많이 속해있다. 본 논문에서는 기업에서 Sales Order - Invoice - Payment - Shipment 의 과정을 수행하는 상황을 특정하였으며, 사용하는 Transaction Data Table 목록은 Table 2와 같다.

Table 2. Transaction Data Table List[3]

Table Name	Comment
Order	Saves Sales Order and Purchase Order information.
Sales_Order_Line	Saves details of the Sales Order Line.
Invoice	Saves invoice information of the Order.
Payment	Saves payment information the client has paid.
Shipment	Saves shipment information of the invoice.

1.3 Master Data

Master Data는 기업이 행하는 다양한 업무에서 기본적으로 다루는 공통된 데이터를 하나의 중앙 데이터베이스로 모아둔 것이다. Master Data를 구축하면 조직 내의 분산된 시스템에서 하나의 공통된 기반 데이터를 사용할 수 있다. 이는 중복된 데이터를 방지하고 다양한 애플리케이션의 View를 통일할 수 있도록 도와주어 업무 효율의 상승을 가져올 수 있다.

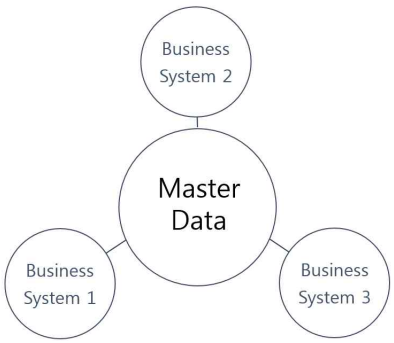

Fig. 5. Business System Integration with Master Data[3]

본 논문에서 정의하는 API는 Sales Order - Invoice - Payment - Shipment의 4가지 UI를 사용 환경으로 가정한다. 보통 Application UI는 많은 테이블 중 필요한 Data를 선택하여 출력하는 방식을 사용하기 때문에 Data Table과 1:1로 대응하지는 않는다. 따라서 4가지의 UI를 구성하기 위해서는 4개 이상의 Table이 필요하며 그 목록은 Table 3과 같다.

Master Data를 구성하는 Table의 선정 기준은 본 논문의 시나리오에서 사용할 Opensource ERP Tool인 Adempiere가 Sales Order - Invoice - Payment - Shipment UI에서 실제로 사용하는 Table 목록을 참고하였다.

Table 3. Master Data Table List[3]

Table Name	Comment
Product	Saves basic information such as product name and ID.
Price_List	Saves price criteria regarding Export/Standard.
Client	Saves Client information.
Business_Partner	Saves Business Partner information.
Product_Category	Saves the Categories of Products.
Product_Type	Saves the Type of the Products.
Warehouse	Saves Warehouse information used by companies.
Currency	Manages Currency information.
Document_Action	Having a value specified for prepare, close etc.
Inventory	Saves and manages the quantity information of Products for each Warehouse.
Organization	Saves information of corporations and organizations.
Delivery_Rule	Specifies delivery methods.
Payment_Term	Specifying payment method such as installment or Immediate payment.
Tender_Type	Specifying payment method such as credit card or cash.
Bank	Saves information of Banks and Accounts.
Account	Saves detailed information of the Accounts
Account_Type	Specifies the Type of Account (established)
Unit_of_Measure	Saves the unit of Product count
TAX	Saves TAX information
Priority	Classifies and Saves Priority information

2. Master Data Management API definitions

2.1 Getter API(Select API)

Master Data에 저장되어있는 값들 중 필요한 데이터를 값으로 반환하는 기본적인 API이다. SQL에서 SELECT문이 수행하는 기능을 가지며, 개발자가 필요한 값을 데이터베이스에 직접 접근하지 않고도 참조할 수 있도록 한다. Getter API는 주로 Spinner 형태의 선택형 UI나 특정 조건을 만족하는 List를 출력하는 UI를 구현할 때 사용한다. API는 Adempiere의 UI에 따라 분류하였으며, 그 목록은 Table 4와 같다.

Table 4. Getter API List[3]

Adempiere UI	Function
Sales Order	• getCategory(category) • getClient(client) • getCurrency(currency) • getDocAction(docAction) • getLocation(location) • getPartner(partner) • getProduct(productName) • getProdPrice(productName) • getWarehouse(warehouse)
Sales Order Line	• getAvailable(pName, wHouse) • getLoadProduct(wHouse, plVer) • getOnhandQty(pName, wHouse) • getOrderedQty(pName, wHouse) • getOrganization(organization) • getQuantity(pName, wHouse) • getReservedQty(pName, wHouse)
Invoice	• getPayterm(payterm) • getTax(tax)
Shipment	• getDeliveryRule(dlvrRule) • getPriority(priority)
Payment	• getBank(bank) • getAccount(client, organization) • setTendType(tendType)
Account	• getCreditAvailable(cli, bank, acc) • getBPartnerGrp(partnerName)

getCategory(category) API를 예로 들면, 해당 API Header는 public String getCategory(String category); 와 같은 형태로 소스코드에 구현한다. 사용자가 Category를 선택 혹은 입력할 경우 이를 파라미터로 받아 Database에 존재하는지 확인한다. 존재할 경우 해당 Category를 String으로 반환하고, 존재하지 않을 경우 사용자에게 Error Message를 반환한다.

2.2 Setter API(Insert/Delete/Update API)

Master Data에 저장되어있는 값을 변경해야할 때 사용하는 API이다. SQL에서 INSERT/DELETE/UPDATE문이 수행하는 기능을 가지며, API를 통한 값의 수정 외의 다른 방법을 차단함으로써 Data의 형식을 틀리는 경우 등을 줄일 수 있다. API는 Adempiere의 UI에 따라 분류하였으며, 그 목록은 Table 5와 같다.

Table 5. Update API List[3]

Adempiere UI	Function
Product	• addProduct(pName, wHouse, pId, avail, price, category) • editBalance(client, account) • editCategory(pName, pId, wHouse, category) • editCredit(client, account) • editInventory(wHouse, pId, pName, avail, onHand, reserved, ordered) • editPrice(pName, prodId, wHouse, price) • editProdName(prName, newName, wHouse) • editProdId(pId, newId, wHouse)
Sales Order Line	• calcAmount(qty, price)
Account	• saveAccount(client, org, bank, acc, currency, accType, crdLim, curBal) • deleteAccount(account) • updateAccount()

addProduct(pName, wHouse, pId, avail, price, category) API를 예로 들면, 해당 API Header는 소스코드상에서 public void addProduct(String productName, String warehouse, string productId, int available, String price, String category)의 형태로 구현한다. 사용자가 새로운 Product를 추가할 때 사용하는 API 함수이다. Product Name, Warehouse, Product Id, Available, Price등을 파라미터로 받아 Database에 존재하는 Product 테이블의 Product List와 비교하여 중복되지 않을 경우 새로운 Product 항목을 추가한다. 중복된 Product Id가 존재할 경우 Error Message를 반환한다.

IV. Expected Result

본 논문에서 제안하는 조립형 ERP 모델이 기존의 ERP와 가장 큰 차이점은 사용자가 필요한 Master Data Module만을 선정한 다음 API를 통해 연결하여 사용할 수 있다는 것이다. 아래의 Fig. 6 ~ Fig. 8을 참고하여 기존의 ERP 모델과 조립형 ERP 모델의 구성방식 차이를 확인할 수 있다.

Fig. 6. Existing ERP module configurations[3]

기존의 ERP 모델은 주로 Package 형식으로 구성된다. 기업에서 최초 도입 시 기업별로 상이한 Business Process에 맞게 수정하는 과정이 필요하며 이 과정에서 많은 비용과 시간이 소모된다. 또한 다양한 독립 Module이 하나의 Package로 묶여 있는 형태이기 때문에 기업의 Business Process가 바뀌게 되면 ERP를 다시 수정하는 것이 아니라 새로 도입하는 수준의 비용과 시간이 소모된다.

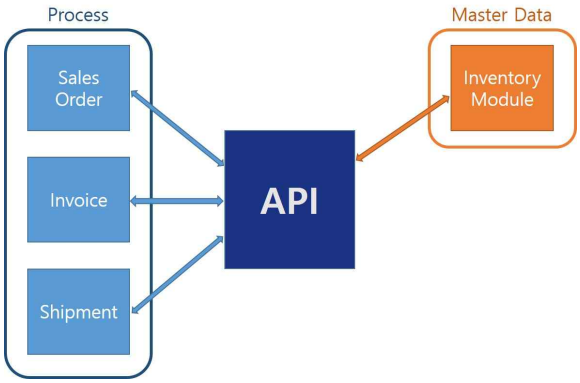

Fig. 7. Configuration of Assembled ERP model

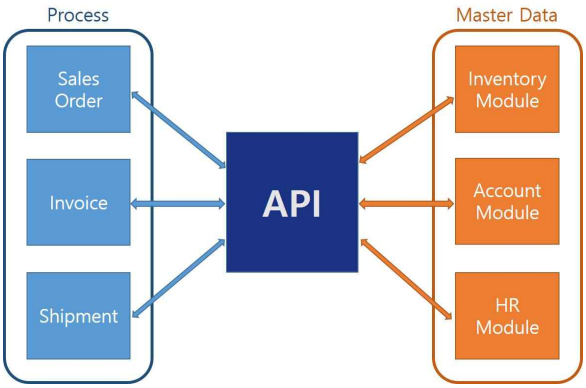

Fig. 8. The way of Adding modules on Assembled ERP

반면, 본 논문에서 제안하는 조립형 ERP 모델은 기업의 개별 Process와 공통 Module을 분리하여 API를 통해서만 연결할 수 있도록 한다. 먼저 ERP 개발을 의뢰받은 개발사는 기업과의 회의를 통해 기업이 자체적으로 개발할 필요가 있는 기업 개별 Process와 공통 모듈을 사용할 수 있는 Process를 분류한다. 다음으로 기업은 소수의 개별 Process 부분에 대한 ERP 개발을 진행하고, 개발사는 필요한 공통 Module을 선별한 뒤 기업에게 API List를 제공하여 선정된 Module을 연결할 수 있도록 한다. 이렇게 이미 시장에서 검증된 Module을 연결하여 사용함으로써 개발 범위를 줄이고 시스템의 신뢰도 향상까지 기대할 수 있다.

1. Scenario

Sales Order Process를 진행하는 과정에서 Order Line을

작성하는 부분을 시나리오로 가정하여 Getter/Setter API가 동작하는 시점을 제시할 수 있다.

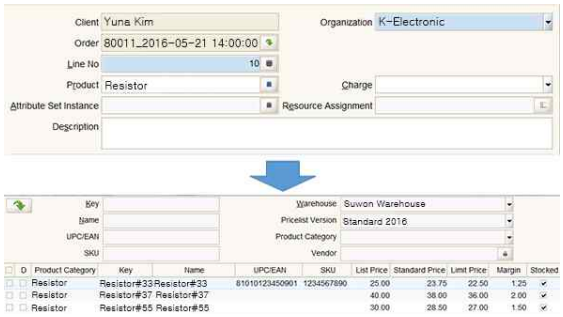

Fig. 9. Example UI of Getter/Setter API

Fig. 9에서 파란 화살표 상단은 Order Line 작성 시 Product를 선택하는 화면이다. 화살표 하단은 사용자가 Product List를 요청함에 따라 API로 연결되어있는 Inventory Module에 존재하는 Product List가 반환된 화면이다.

사용자가 Product List를 요청하는 동안 Sales Order Process와 Inentory Module 사이에서 public ArrayList <String> loadProduct(String warehouse, String plVersion) 이라는 Getter API가 동작한다. 호출된 API는 사용자가 선택한 Warehouse와 Product List Version 값을 Parameter로 받아 Inventory Module로 전달하고, Inventory Module은 조건에 맞는 Product List를 반환한다.

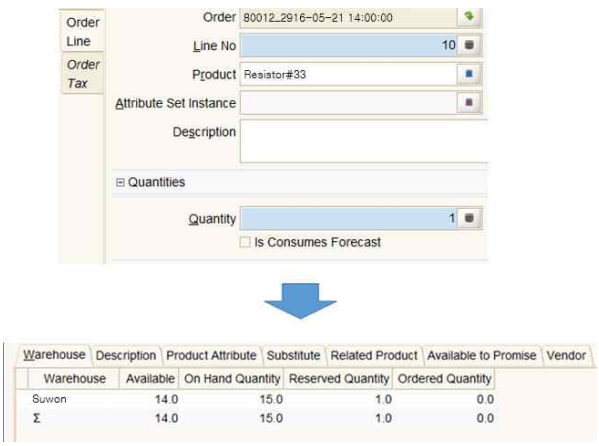

Fig. 10. Example UI of Update API

Fig. 10에서 파란 화살표 상단은 사용자가 Product를 선택한 다음 Sales Order의 Status를 Prepare로 전환하는 것을 보여주는 화면이다. 화살표 하단은 사용자가 작성한 Sales Order의 Status를 바꿈에 따라 Inventory Module에 존재하는 Product의 개수가 변하는 것을 확인할 수 있는 화면이다.

이 과정에서 동작하는 API는 public void prepareOrder (String warehouse, String pid, String pname, int orderedQuantity)라는 Setter API이다. 사용자는 Sales Order를 작성하면서 Elm_Elm Tree라는 Product를 1개 선택한 다음 이 Order의 Status를 Prepare로 바꾸었다. 이에 API는 사용자가 선택한 Product의 Id, Warehouse, Name, 그리고 주문에 포함된 Quantity의 정보를 Parameter로 하여 Inventory Module로 전달하고, Inventory Module은 전달받은 정보를 바탕으로 해당 Product의 Available, On Hand Quantity, Reserved Quantity, Ordered Quantity의 값을 계산하여 저장한다.

2. Comparison and Analysis

Table 6은 ERP 모델을 Package 구입, Package 개발, API 개발(제안 방식)으로 구분하여 간략하게 나타낸 것이다.

Table 6. Comparison between the ERP Models[3]

Kinds / Category	Package Purchase	Package Development	Assembled ERP
Required Time	Build+Education (Average : 6~8months)	Full Package Development	Part of Package Development+ API Development
Cost	Purchase Package, ERP Consulting, Build and Education, DBMS License, Maintenance, Operating, Add admin account	Labor Cost, DBMS License	Labor Cost, DBMS License, 3rd-Party App Purchase
Business Suitability	High Customizing Cost required.	Developing for Business purposes	Developing for Business purposes
Remarks	Burdensome costs for small businesses.	Increase of labor costs due to increased development period.	Requires agreements with 3rd-party App Development Company

본 논문에서 제안하는 API 개발 방식은 Package 개발 방식과 비교하여 개발해야 할 항목이 현저히 줄어들기 때문에 시스템 도입에 필요한 총 소요 시간이 짧다는 점에서 가장 우세한 모습을 확인할 수 있다. Package 구입과 비교해도 비용이 들어가는 항목이 적고, 업무 적합성이 높아 효율성이 좋다고 평가할 수 있다.

특히 ERP를 도입한 소규모 기업을 대상으로 한 설문 조사 결과 패키지 ERP의 기능 축소, 업무 처리 절차 단순화를 위한 커스터마이징이 성공적인 ERP구현에 큰 영향을 주는 것으로 나타났으며, 이는 공통 분야에서 최소한의 패키지 형식의 ERP를 사용하고 특징적 업무 분야에서는 직접 ERP를 개발하여 사용함으로써 ERP의 기능을 축소하고 데이터의 복잡도를 줄인 조립형 ERP 모델이 중소기업에 적합한 개발 모델이 될 수 있음을 보여준다고 할 수 있다.[10]

하지만 시스템을 구축하기 위해서 Third-Party Application을 사용해야 하는데, 사용자가 원하는 Application을 개발한 사측에서 API를 사용할 수 있도록 UI를 구성하는 등 협조를 해야만 시스템에 포함할 수 있다는 점에서 연결할 수 있는 Application의 다양성을 확보하기가 어려울 수 있다는 점이 단점이 될 수 있다.

3. Advantages

조립형 ERP 모델을 사용했을 경우 기대되는 효과는 크게 세

가지로 나타낼 수 있는데, 개발 효율 증가, 개발 및 사용 난이도 하락, 그리고 중복 Table 예방이다.

첫째, 개발 효율측면에서 ERP 시스템을 도입하는 과정에서 개발자와 사용자는 ERP에서 사용하는 Table의 구성과 각 Data의 형식 등 많은 부분에 대해 알고 있어야 한다. 하지만 조립형 ERP 모델을 사용할 경우 기업에 필요한 Process Database에 대한 지식만 사용하여 기업에서 필요로 하는 부분에 대한 개발하면 되기 때문에 개발 시간 및 투자비용이 줄어들어 개발 효율을 향상시킬 수 있다.

둘째, 보통 ERP 시스템을 도입할 때 Master Data 부분에 해당하는 Module을 직접 개발하고 Database를 구축하려면 각 분야에 대한 깊은 지식이 필요하다. ERP를 도입할 때에는 Database Modeling만 하는 것이 아니라 ERP 내에서 각각의 Data가 갖는 의미를 파악하고, 어떤 논리적 관계를 맺고 있는지 등을 모두 고려해야하기 때문에 단순히 상황에 맞는 API를 찾아 연결해주는 작업보다 개발 및 사용 난이도가 높다고 할 수 있다. 이를 해결하기 위해 각 분야에 해당하는 Module을 직접 개발하지 않고 API를 통해 Third-Party Application을 연결하여 필요한 module을 구입하고 API를 호출하여 사용하는 방식을 사용한다면 개발자는 물론 ERP 시스템 사용자도 각 분야에 대한 고급 지식이 부족하더라도 쉽게 ERP 시스템을 구축하고 사용할 수 있다.

셋째, Database중복 이슈는 Database를 사용하는 애플리케이션에서 필연적으로 발생할 수밖에 없는 일이다. 특히 ERP 시스템을 구축한 뒤에 Business Process가 크게 바뀌거나 업무가 확장되는 일이 발생하면 Database에 새로운 Table이 추가되거나 기존 Table을 변경하게 되는데, 이렇게 기존 Database를 유지하면서 Table을 수정하다 보면 같은 내용을 가지는 Table이 생길 가능성이 높아지고 이는 Database의 신뢰도와 퍼포먼스를 하락시키게 된다. 하지만 본 논문에서 제안하는 방식은 업무가 추가되더라도 추가된 업무에 해당하는 Database를 개발하지 않고 필요한 Database를 포함하는 Module을 추가하여 바로 실무에서 사용할 수 있으며, API를 통한 간접적인 Database 접근만을 허용하기 때문에 사용자 임의로 Table을 수정하거나 생성할 수 없도록 하여 업무의 확대나 변경 시 발생할 수 있는 Table의 중복 이슈를 최소화할 수 있다.

4. Characteristic

4.1 Reliability

ERP Package를 사용할 경우 기업은 Package가 자신들의 Process에 부합하는지에 대한 고민과 더불어 Master Data에 해당하는 각 분야의 Database의 Data를 관리하는 Logic이 어떤 방식으로 동작하는지에 대한 고려도 해야 했다. 예를 들면 Master Data에 해당하는 Account의 경우 Sales나 Purchase의 진행에 따라 발생하는 사용자 계정별 잔액 계산, 신용과 현금의 결제 비율 조정 등의 Logic이 존재하고, Inventory의 경우 Warehouse별 재고 조정, Product의 판매나 예약 등 Status의 변화 등의 Logic이 필요하다. 이러한 분야별 Logic의 완성도는 ERP 시스템의 신뢰도에 직접적으로 영향을 주게 된다. 따라서 ERP 시스템의 신뢰도와 완성도를 높이기 위해서는 Database에서 사용하는 Logic이 검증되어야 하며 높은 신뢰도를 갖는 검증된 Database Logic을 개발하기 위해서는 Account나 Inventory등 해당 분야에 대한 깊은 이해가 필요하다. 또한 도입 이후에도 취약점 보완을 위해 많은 시간과 노력을 투자해야 하므로 전체적인 개발 및 시스템 구축 기간이 증가하게 된다.

이를 해결하기 위해 검증이 끝난 완성된 ERP Module을 사용할 수 있다. 시중에는 Module 형태의 ERP가 존재하며 이러한 ERP는 사업 분야에 따라 필요한 Module을 선택하여 구입이 가능한데, 예를 들어 건축업이라면 자재 관리 특화, 인사는 사원 정보 관리 특화 Module을 구입하여 사용할 수 있다. 따라서 구매나 판매, 급여, 승진과 같이 회사마다 차이점이 존재하는 부분은 직접 관리 Module을 개발하고, 재정 관리나 자원 관리처럼 대부분 사업에서 공통적인 부분에 해당하는 Module은 이미 시중에서 사용되며 검증이 끝난 Third-Party Application을 API로 연결하여 ERP 시스템을 구축할 경우 각 Module에 대한 신뢰도가 보장되기 때문에 ERP 시스템의 신뢰도 또한 높아지게 된다. 추가로 시스템 구축 기간 또한 크게 단축할 수 있다.

4.2 Expandability and Flexibility

조립형 ERP는 기업의 업무가 추가되어 새로운 Module이 필요할 경우 필요한 Module을 새로 구입하여 API를 통해 연결만 해주면 바로 업무에 적용할 수 있으므로 확장성이 뛰어나다. 단, 사용하고자 하는 Third-Party Application이 API와 그에 해당하는 UI를 지원해주어야 하며, 지원만 된다면 제조사에 관계없이 다양한 Module을 사용할 수 있으므로 많은 업무분야에 확장하여 적용 가능하다.

또한 업무의 추가뿐만 아니라 기존 업무의 Process에 변동이 있는 경우에도 사용자의 요구사항에 유연하게 대처할 수 있다는 점이 기존 ERP Package와 차별되는 점이다. 예를 들어 처음 도입할 당시에는 고려하지 않았던 Data에 접근해야 하는 경우가 생길 수 있다. 이러한 상황이 발생할 경우 사용자가 요청하는 Data에 해당하는 Table Name, Column Name, Value를 Parameter로 받는 Getter/Setter API를 추가하여 사용자가 필요로 하는 Data를 사용할 수 있도록 한다. 이처럼 ERP 시스템 도입 후에도 Module 제공 업체와 협의하여 Update나 Patch를 하는 방식으로 사용자가 추가적으로 요구하는 API를 제공하여 시스템의 유연성을 향상시킬 수 있다.

5. Trade-Off Relation

조립형 ERP를 이용해 시스템을 구축할 때 사용자는 개발비용과 사용자 요구사항 충족 사이에서 어느 쪽에 비중을 둘 것인지 선택을 해야 한다. 사용자가 요구하는 기능의 API를 모두 구현하면 개발기간과 개발 비용이 증가하게 되며, 비용 감축에

비중을 둘 경우 사용자의 요구사항과는 동떨어진 부족한 시스템이 만들어지게 된다. 즉 개발 기간과 개발 비용은 API 정확도와 Trade-Off 관계에 있는 것이다. 따라서 조립형 ERP 시스템을 도입할 때 개발 비용과 API 정확도 향상 사이에서 적절한 비율을 찾아야 하는데, 이를 결정하는 과정을 두 가지 단계로 나누어 생각해볼 수 있다.

먼저 Third-Party Application을 제공하는 업체가 보유한 자료를 활용하여 API의 수준을 결정한다. 이미 상업화에 성공한 Application을 운영하고 있는 기업이라면 그 Application이 지원하는 분야에서 사용자들이 주로 요구하는 Data와 Table 구조에 대해 축적해 둔 정보가 있을 것이기 때문에 이를 기준으로 하여 보편적으로 받아들일 수 있는 API의 기능 수준을 설정할 수 있다.

다음으로 이 기준을 가지고 ERP 시스템을 구축하려는 사용자와 협의하여 기준에서 기능 추가에 더 무게를 둘 것인지, 혹은 기능을 제외하고 더 가볍게 디자인 하여 비용을 줄이는 것에 중점을 둘 것인지를 결정하는 방법으로 사용자의 요구사항을 반영하는 것이다.

V. Conclusions

기존에도 다양한 Business Application을 통합하여 ERP 시스템을 구축하려는 시도는 있었지만, 대부분 중심이 되는 Application을 개발한 기업의 패키지 상품사이의 연결에 집중되어 있는 것이 그 한계로 지적되었다. 또한 기업의 업무 Process와 상관없는 기능을 가진 Module도 Package라는 명목 하에 구입을 하게 되고, 새로운 기능을 추가하기 위해 필요한 시간과 비용이 크다는 것도 중소기업 입장에서는 ERP 시스템 구축을 망설이게 되는 원인이었다.

기존 방식의 단점을 극복하기 위해 본 논문에서는 Package 형식의 ERP 시스템이 아닌 API를 활용한 조립형 ERP 모델을 제안하였다. ERP Package를 Transaction Data(Process)와 Master Data로 나누고, Master Data를 하나의 업무 단위마다 Module로 분류하여 Process와 Module 사이에 API를 두어 복잡한 업무에 필요한 각각의 Module을 추가하여 ERP 시스템을 구축하는 방식이다. 이 방식을 이용하면 각 기업은 기업만의 특색 있는 Business Process에 해당하는 Database와 ERP UI는 직접 개발하고 대부분의 기업이 공통적으로 사용할 수 있는 Master Data는 필요한 분야에 해당하는 Module을 API를 통해 연결하여 기업의 업무 환경에 잘 맞는 ERP 시스템을 완성할 수 있다. ERP 시스템의 크기가 작아지므로 개발에 필요한 비용과 시간이 크게 줄어들고, 맞춤형 개발이기 때문에 기업의 업무에 잘 맞으므로 시스템 구축 실패에 대한 위험 부담 역시 줄어든다. 또한, Master Data의 Module은 모두 독립적으로 존재하기 때문에 시스템 구축 이후에도 사용자의 요구사항에 따라 Module을 추가하거나 삭제할 수 있으므로 유연한 업무 변경과 확장이 가능하다. 이러한 장점들이 그동안 ERP 시스템을 도입하는데 부담을 느꼈던 중소기업들의 진입 장벽을 낮추고 안정적으로 시스템을 구축할 수 있도록 하여 기업의 성장에 기여할 수 있을 것이다.

REFERENCES

[1] “Analysis of the ERP Benefits of SMEs”, Ministry of Trade, Industry and Energy, Press Release, November 2001.

[2] Jong-suk Lee, “The Development Process and Direction Suggestion of ERP(Enterprise Resource Planning)”, Journal of Advanced Information Technology and Convergence(JAITC), Vol. 6, No. 3, pp. 192-199, June 2008.

[3] Jin-kyung Park, “Master Data Management API for ERP Development”, UNIVERSITY OF SEOUL, 2016

[4] Dong-Pyo Seo, “An Empirical study on the Effect of the SMEs ERP Systems Implementation”, HANSUNG UNIVERSITY, 2010.

[5] “The Survey for Domestic Enterprises’ IT Utilization” Ministry of Science, ICT and Future Planning, Press Release, February 2015.

[6] Miri Park, “Design and Implementation of the interface standardization for solution of Enterprise Application Integration”, EWHA WOMANS UNIVERSITY, 2002.

[7] Dong-cheol Sim, “The EAI concepts and Trends in Domestic and Foreign markets”, KISDI IT FOCUS, Vol. 2001, No. 10, pp. 49-52, October 2001.

[8] Yong-seok Seo, “A Monitoring Method for Hub&Spoke-based EAI Systems”, KOREA UNIVERSITY, 2011.

[9] Gwang-Rok Kim, “Design of Maritime Logistics IP-RFID Service Platform API Based on XML”, DONG-A UNIVERSITY, 2013.

[10] Kwang-Seog choi, “A Study on ERP Adoption and Operations in Small and Medium Enterprises”, CHONNAM NATIONAL UNIVERSITY, 2007.

[11] Young-Min Lee, Sang-Ho Ju, “A Case Study on ERP Implementation for e-Business”, Journal of the Korea Society of Computer and Information, Vol. 7, No. 2, pp.187-195, June 2002.

[12] Seung-Gweon Kim, Seung-Bong Park, Jae-Young Kim, “A Empirical Study on ERP Package in Korean SMEs”, Journal of the Korea Society of Computer and

Information, Vol. 12, No. 1, pp.243-252, March 2007.

Authors

Jin Kyung Park received the B.S., M.S. degrees in Electronic and Computer Engineering from University Of Seoul, in 2014 and 2016, respectively.

Park joined the undergraduate researcher of the Database Laboratory at University Of Seoul, Seoul, Korea, in 2012. He is currently a researcher in the Database Laboratory, University Of Seoul. He is interested in Database Management, Internet of Things and Embedded System.

Mi-Su Gim received the M.S. and Ph.D. degrees in Dept. of Electronic & Electrical Engineering from Univ. Of Seoul, Korea, in 2006 and 2015, respectively.

Dr. Gim is currently a researcher in the Database Laboratory, University Of Seoul.
She is interested in Database Management, Internet of Things and Embedded System.