Software Evolution in the Context of .NET Framework

Walid Rohaimi
COMPUTER ENGINEERING
Software Evolution in the context of .NET Framework

Walid Rohaimi

Detta examensarbete är utfört vid Ingenjörshögskolan i Jönköping inom ämnesområdet datateknik. Arbetet är ett led i teknologie magisterutbildningen med inriktning informationsteknik. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Handledare: Vladimir Tarasov
Examiner: Kurt Sandkuhl
Omfattning: 20 poäng (D-nivå)
Arkiveringsnummer:
Abstract

This paper discusses the process of software evolution and especially software migration in the context of .NET Technologies. Actually most of the companies that uses legacy systems implemented with procedural languages as C, Visual Basic and so on, meet some problems when new requirement specifications have to be integrated.

One possibility to deal with this situation is to choose a good migration strategy from these legacy systems towards new Object Oriented design.

There are some migration processes that enable the fulfilment of this task but most of the time theses processes cannot be applied directly without any modification.

This report presents a migration strategy and migration process applied for a real case of an application in a company. The New Object Oriented design of the application and the result are discussed in the following sections of this document.
I would like to thank all people who helped me with my master thesis work

First I would like to thank my parents who give me all their love and affection for ever.

I would like to thank my Teachers at Jönköping School of Engineering especially Vladimir Tarasov and Kurt Sandkuhl.

I would also thank all my colleagues at Schur Packaging Systems who gave me all their attention and especially the Manager of the Computer Systems Department Lars Schön and my colleagues Peter Kessel, and Conny Norin.

Finally but not last, a big thanks to all my friends from all over the world at Jönköping University with whom I spent a very good time.
Acknowledgements

Key words

Software Evolution, Software Migration, .NET Technologies, Strategy of Software Migration, XML Integration Engine.
## Contents

1 Introduction ............................................................................................................. 1

1.1 BACKGROUND .................................................................................................... 1

1.2 PURPOSE/OBJECTIVES .................................................................................... 1

1.3 LIMITATIONS .................................................................................................... 2

1.4 THESIS OUTLINE ............................................................................................. 2

2 Theoretical Background ....................................................................................... 3

2.1 SOFTWARE EVOLUTION .................................................................................. 3

2.1.1 The process of Software Evolution ................................................................. 4

2.1.2 Lehman’s Laws ............................................................................................. 5

2.1.3 Challenges in Software evolution .................................................................. 6

2.2 FROM LEGACY SYSTEMS TO OBJECT ORIENTED PLATFORMS ................. 8

2.2.1 Static Analysis of legacy code [1] .................................................................. 9

2.2.2 Decomposing non batch program [1] .............................................................. 10

2.2.3 Object Model abstraction [1, 8, 9, 10] ......................................................... 11

2.2.4 Reengineering phase [1, 11, 12] .................................................................. 11

2.2.5 Encapsulating the identified object into wrappers [1] ................................. 11

2.2.6 Incremental translation of object wrappers [1] ............................................. 11

3 Practical Part ........................................................................................................ 12

3.1 PROJECT DESCRIPTION ................................................................................. 12

3.1.1 Overview of the production system .............................................................. 13

3.1.2 Context diagram .......................................................................................... 14

3.2 MIGRATION PROCESS .................................................................................... 16

3.2.1 Requirement Specification Phase ................................................................. 18

3.2.2 Design Phase ................................................................................................ 22

3.2.3 Implementation Strategy ............................................................................. 27

3.2.3 Test phase ...................................................................................................... 28

4 Results .................................................................................................................. 29

5 Perspectives ........................................................................................................ 32

5.1 THEORETICAL PERSPECTIVES .................................................................. 32

5.2 TECHNICAL PERSPECTIVES ....................................................................... 32

5 Conclusions ......................................................................................................... 33

7 References ......................................................................................................... 34

8 Appendixes ........................................................................................................ 36
List of Figures

FIGURE 1 ..............................................................................................................4
FIGURE 2 ..............................................................................................................6
FIGURE 3 ..............................................................................................................9
FIGURE 4 ............................................................................................................10
FIGURE 5 ............................................................................................................13
FIGURE 6 ............................................................................................................15
FIGURE 7 ............................................................................................................17
FIGURE 8 ............................................................................................................19
FIGURE 9 ............................................................................................................20
FIGURE 10 ..........................................................................................................21
FIGURE 11 ..........................................................................................................23
FIGURE 12 ..........................................................................................................25
FIGURE 13 ..........................................................................................................26
1 Introduction

1.1 Background

Nowadays the research area of software evolution becomes a big deal for most of research groups in software development area and IT departments in companies.

The actual software applications should be constantly changed or updated in order to follow the evolution of the business orientation and the business needs.

In order to successfully evolve the actual applications in companies, software developers should make a real project plan for that, supporting certain phases of the software evolution process according to the nature of changes they want to apply.

In fact, most of the time the principal evolutions of software applications are due to the emergence of new functional requirements claimed ordered by the stakeholders or the users, and this leads either to an adjustment of the actual application or a migration from legacy systems towards new platforms.

1.2 Purpose/Objectives

The main purpose of this project is to present a good way to fulfil the task of migrating from an old application, based on procedural language towards a new Object Oriented system.

The objective is to find the best strategy for a real case in a company where the actual source code is not designed to fit with new system requirements and also with an Object oriented design.
1.3 Limitations

This project will only focus on the study of the migration process from an old application implemented by functional languages toward new Object Oriented application the case study applied for is in the domain of machine controllers and data exchange between a planning system and a centralized database in a factory.

1.4 Thesis outline

The first part of this work is a presentation of the theoretical background which contains a state of art of the actual software evolution processes, the challenges and perspectives within this area and also a description of a framework for software migration.

The second part of this document is a case study in a company which has as an objective to perform the migration from a procedural application implemented using Visual Basic 6.0 towards a new version implemented in .NET platform including new functional specifications.

Parts 3 presents the results of the work described previously in part 2 with a description of advantages and perspectives of the new application.

Finally, part 4 will be a conclusion of the work including some remarks and new ideas for the research area of software evolution.
2 Theoretical Background

2.1 Software Evolution

The term evolution is used in different area of science and society such as natural sciences, sociology history and so on. Entities, concepts theories and ideas always follow a cycle of a continued progression and transformation of their composition and properties. This process of change in one or more of the class attributes leads to the emergence of new properties or to improvement, in some sense. In general, the change will be such as to adapt the elements of the class so that they maintain or improve their fitness to a changing environment. [14]

The change can be either positive or negative depending on its purpose and who this change has been made. In the positive way, this change enable to an entity to be more useful and meaningful by increasing its value and worth.

Alternatively or at the same time, evolution of the class may remove properties no longer appropriate. Changes are generally incremental and small relative to the entity as a whole but exceptions may occur. [14]

In the area of software development and information systems the definition of this process could be as follow:

“Software Evolution is the process by which program change shape, adapt to the marketplace and inherit characteristics from pre-existing evolution.” [9].

Nowadays the process of software evolution became inevitable in most of the companies. The importance of this process is due to the emergence of new requirement when the software is used, the evolution of the business environment and the need for improving performance flexibility and reliability of the actual system.

Thus, In order to maintain the value of the actual software to the business organization, they must be changed, updated or migrated towards a new systems or platforms.

In this section of the report we will give an overview of Software Evolution process, a presentation of Lehman’s Laws, which have a big contribution on Software Evolution study and finally an introduction of new challenges in the area of Software Evolution.
2.1.1 The process of Software Evolution

The use of a software evolution process depends on many internal and external factors. Among them we can have:
- The type of software being maintained
- The development process used
- The programming language used for implementing the actual software (Procedural or Object Oriented languages).
- The skills and experience of people involved.

The following diagram presents a framework for software Evolution process.

---

**FIGURE 1. FRAMEWORK FOR SOFTWARE EVOLUTION PROCESS[10]**
According to this diagram we notice that change requests - formulated by stakeholders and users - have to be analyzed earlier so that we can make a decision on which strategy to choose and what kind of changes have to be made (For instance: Fault repair, Platform adaptation or system enhancement [10]).

Then changing implementation of the actual software can start, and finally a new version of the software can be released.

The diagram also shows that software evolution process is iterative so that we can restart it when new change requests are formulated.

The phase of changing implementation is mostly based on System Re-engineering activity which enables restructuring or re-writing part or all of a legacy system without changing its functionality.

Re-engineering involves adding effort to make software easier to maintain, and the advantage of such activity is the reduction of risks and costs since it is significantly less than the cost of developing new software.

2.1.2 Lehman’s Laws

The work of Lehman’s in the area of software development and software maintenance has an important effect in the way to introduce Software evolution into Software development process.

For Lehman, the place to look when using software evolution process is within the development process itself – Software development and evolution should be a single iterative process – and a system views as feedback and biased toward increasing complexity [9].

After major empirical studies, Lehman proposed a number of “Laws” which represent insights into system evolution and have to be taken into account during the process of software evolution.
2.1.3 Challenges in Software evolution

The domain of Software evolution still undiscovered and unevaluated in traditional software development process because it has to be integrated in the middle of it and taken into consideration from the beginning. Thus some new needs in the research area of software evolution have to be identified in order to increase the possibilities of better integration of software evolution process.

Here we introduce some of the challenges for software evolution made by Chase 2005 and classified as following:

- Time horizon
- Research target
- Stakeholders
- Type of artefact understudy
- Type of support needed

**FIGURE 2. LEHMAN’S LAWS [9].**
Enumeration of challenges [6]:

- Preserving and improving software quality
- A common software evolution platform
- Supporting model evolution
- Supporting co-evolution
- Formal support for evolution
- Evolution as a language construct
- Support for multi language systems
- Integrating change in the software life cycle
- Increasing managerial awareness
- Need for better versioning systems
- Integrating data from various sources
- Analyzing huge amount of data
- Empirical research
- Need for improved predictive models
- Evolution benchmark
- Teaching software evolution
- Post deployment runtime evolution
2.2 From Legacy Systems to Object Oriented Platforms

The term migration originates from the Latin word “migrate” which means “to go from one place to another”. It’s used in sociology and biology to describe a slow but steady movement of a population from one place to another. [17]

In computer science it is used in “data migration” or “hardware migration”. The latter means switching from one hardware platform to another one. Data migration is the process in which the structure of a database is changed and the stored data adjusted to the new structure. [16]

Nowadays most of the companies choose to migrate from their legacy systems towards Object Oriented Platforms. The main reason of that is the ability of those systems to keep competitive by reusing components, building open systems and having applications that fit with their business organisation and business processes.

A change in a programming language almost always implies a change in the architecture of the software system. This is especially true when changing from one type of programming language to another, e.g. from a script language to a higher object oriented language.

However, this decision is not always that easy to take, since the existing systems have been implemented with large investments and also because they are considered as a real knowledge base in to the company, storing the expertise of both developers and users.

Another aspect that can affect the decision of migration is whether the company has enough investments and human resources to allocate for building new Object Oriented applications from scratch, following the process of software development (Requirement specification -> Design phase -> Implementation -> Test -> Deployment).

The migration towards Object Oriented entails the definition of an Object Oriented model of the target application domain and their relationship [1].

An alternative approach to the problem of designing an Object Oriented model of the new application is by reverse engineering the existing system [1].

The Object Oriented model will give an overview of the whole system and should contain all the actions the user can perform with the system, but also new actions written as requirements specifications by the client.
One way of building this Object Oriented model is to use a framework for migrating procedural code mostly used in legacy systems towards Object Oriented. This framework is based on an incremental and iterative migration process where legacy procedural source code is reengineered [2].

The activity of designing an Object Oriented model is a part of a general process of migration. This process consists of six sequential phases assisted with a framework for source code representation.

**FIGURE 3. PROPOSED PROCESS FOR MIGRATING PROCEDURAL SYSTEMS TO OBJECT ORIENTED PLATFORMS[2]**

### 2.2.1 Static Analysis of legacy code

This phase consists of analysing source code and extracting useful information from it. These information can be stored either in relational database [1] (Both fine grained information at and coarse-grained information at intraprocedural level) or in repository [3].

The framework proposed in [2] allows for procedural source code to be represented at higher level of abstraction.

A good way for source code representation is to use a portable model called *Abstract Syntax Tree AST* [2] based on XLM structure.
In fact this AST is a document that contains relevant information about the source code of the program.

Two approaches are used to map AST into XML:
- Bottom up
- Top down

Both of those approaches will result on a DOM for representing the tree and a DTD describing this document.

```
<EXPRESSION-STATEMENT>
  <EXPRESSION-STATEMENT-BODY>
    <ASSIGNMENT surface-syntax="shuffle_level = num_decks * 26">
      <ASSIGNMENT-TARGET surface-syntax="shuffle_level">
        <IDENTIFIER-REF id-name="shuffle_level"/>
      </ASSIGNMENT-TARGET>
      <ASSIGNMENT-SOURCE surface-syntax="num_decks * 26">
        <MULTIPLICATION surface-syntax="num_decks * 26">
          <IDENTIFIER-REF id-name="num_decks"/>
          <INT-LITERAL int-long="NIL" int-radix="10"
          int-unsigned="NIL" value="26"/>
        </MULTIPLICATION>
      </ASSIGNMENT-SOURCE>
    </ASSIGNMENT>
  </EXPRESSION-STATEMENT-BODY>
</EXPRESSION-STATEMENT>
```

**FIGURE 4. XML ELEMENT FOR EXPRESSION STATEMENT IN C [1]**

### 2.2.2 Decomposing non batch program

A reverse engineering process is performed [1]. During this phase we try to identify the functions and data that can be chosen for implementing the client application and the server side.

To fulfill this task, some useful algorithms can be used:
- System segmentation algorithm [4]: this algorithm produces clusters that assemble maximum source code properties related to a class candidate.
- Clustering algorithm [4]: This algorithm takes the ASTs as input and produces clusters represented by sequence of tuples including functions and data types.
2.2.3 Object Model abstraction

This phase aims to identify the object model by applying a reverse engineering activity. One way to deal with this phase is to extract model by a series of an iterative analysis steps applied at AST level [2].

The process of extraction focuses on the analysis of global aggregate data types and formal parameters lists [2].

2.2.4 Reengineering phase

The user interface components (Client side) and the methods of the object (Server side) identified in the previous phase are encapsulated in different programs [3].

This will enable the designer to reuse certain components of the legacy system during the implementation phase of the Object Oriented system.

The method of encapsulating functions into objects is called “Wrapping Technology”: The identified “Legacy Objects” are encapsulated into wrappers and new Object Oriented systems use existing resources through the wrapper’s interface [1].

2.2.5 Encapsulating the identified object into wrappers

Objects identified in the abstraction step are encapsulated into wrappers. Each group of programs (represented by ellipses) and persistent data stores implementing an object is encapsulated into an object wrapper. The wrapper interface provides a method (represented by a rectangle on the edge of the wrapper) [13].

2.2.6 Incremental translation of object wrappers

This phase consist of decoupling the component being translated from the other components. It involves impact analysis and maintenance operations in order.
3 Practical Part

3.1 Project Description

The aim of this project is to apply some concepts of software evolution presented in the theoretical part - described in the previous section of the report- to a real life case study in a company.

This case study is about designing and implementing a new version of an actual application called XML Integration Engine XIE. This application handles the parsing and generation of XML documents exchanged between a production system and a centralized database.

In fact the application should work automatically in order to parse data coming form the planning system to the central database. This database is requested by machines on the production area in order to put bundles on pallets. The coming XML messages contain either a daily distribution plan for producing bundles and putting them into pallets or an urgent message for the palletiser in order to start or stop producing a specific product.

The new application should also work in manual mode. By the mean of its GUI, the user can parse and generate XML documents when XIE having problems in the automatic mode, and information about parsed and generated documents should be displayed as well as errors and other events and configuration setups.

The generated XML files are sent to the production system tat store all information about produced pallets and other information regarding the trucks and conveyor system on the production area. This information is later on used to build reports for the production manager with all relevant fields from the database that have been generated into XML documents.

The actual version of XIE handles data exchange by the mean of IBM MQ Series message queue. The message queue is accessed from XIE via MQ Series API.

XIE is based on MSXML4 from Microsoft and coded in Visual Basic 6.0.the following sections gives an overview of the Production system at Schur Palletizer Systems to be able to clearly identify the position of XIE and its importance for the whole production system. We also give a presentation of the actual version of XIE by the mean of a Context diagram.
3.1.1 Overview of the production system

This figure gives an overview of the production system. This system has a centralized SQL Server database called TMC and XIE can access to it.

XIE has a connection with SAM system through reception and transmission queues.

FIGURE 5. OVERVIEW OF THE ACTUAL INFORMATION SYSTEM
3.1.2 Context diagram

The following figure illustrates the overall design of the XML Integration Engine. The program consists of:

- **XMLSupervision**: The main supervision program which instantiates the Class Objects and provides a user interface for monitoring the XIE.

- **ObjXMLFilter**: An instance of a class calling all the methods for filtering received messages. The filtering process can be handled from either a queue (IBM queue or MS queue) or a folder where the messages can be put for testing the application manually. The time critical message ‘TieLine Assign’ is conveyed to the internal queue Int1 for immediate processing whereas non-time critical messages are transferred to the internal queue Int2.

- **Parser 1**: An instance of a class polling the internal queue Int1. Messages are parsed and the information stored within [XMLInBox1].

- **Parser 2**: An instance of a class polling the internal queue Int2. Messages are parsed and the information stored within [XMLInBox2].

- **ObjXMLGenerate**: An instance of a class polling the [XMLOutBox] for Requests and Production Data to SAM. New data is processed into XML Documents and stored on the Transmission Data Queue – TO.SAM.DATA. This object uses XML Schema for creating XML Documents so that we can have as many representations as we want according to available XML Schemas.
On start up, the application starts on the default mode by automatically initializing all the instances of classes described previously. However, the User can change this default mode in case of troubles into manual mode by setting the configuration file of the application. He can also choose between two types of GUI of the application, either the one with data reception and transmission queues based on IBM queue or the other one based on MS queue.
3.2 Migration process

After reading articles and references about state of art concerning Software evolution and specially the domain of software migration processes described in the previous sections, a decision of choosing a relative migration process for the actual project had to be taken.

In fact, the previous version of XIE was implemented in Visual Basic 6.0 and one of the needs from the clients was to perform its migration towards C# which is an Object Oriented programming language.

The framework for migration from legacy system towards Object Oriented Platform seems to be useful in our case study.

However, the problem with this solution is that the process of Re-engineering the whole previous version of XIE and then building an Object Model based on it is very time consuming especially if we want to use ASTs representation and software goal graph dependency…etc.

Also, these kinds of processes are well defined for big applications that need a big level of formality and methodology which is not the case of our application.

The migration process presented bellow takes some relevant parts of the migration framework described in the previous chapter, like Re-engineering process and source code analysis and also Object Model definition and integrate them into a classical software development process which contains requirement specification phase, design phase, implementation phase and finally test phase with an iterative back up during those phases.
The following part describes more in detail each step of this migration process used for this project.

3.2.1 Requirement Specification Phase

1/ Reengineering process and Source code analysis

This step consists of reading the source code of the previous version of XIE implemented in VB6. The project folder of XIE contains three sub folders:

- XIE Common: this sub folder contains source code enabling Data Source access, the management of IBM queues and the code for log file handling.
- XIE ActiveX contains modules enabling the parsing, generating and filtering XML documents.
- XIE Client folder contains source code for user interface application and code for accessing registry file.

After reading all the source code contained in these 3 folders, I started extracting the relevant information according to the context diagram given in XIE documentation.

For each action that the user can perform using XIE, I classified the important variables, Data Types and functions and tried to group them into classes in order to have a first version of an Object Model.

I have also chosen to group those important information based on the 3-tiers architecture model as described in the following figure:
In this section we will present a list of new requirement specifications that the stakeholders want in the new version of XIE. This list is also based on the XIE documentation available at Schur Packaging Systems [7].

The figure below shows these requirements which are described in this case as tasks to be fulfilled. They are ordered by their level of priority to be able to assess their efficiency later on the Test phase.

We also assign a type of each task in order to group them into categories of changes measurements.
<table>
<thead>
<tr>
<th>Priority order</th>
<th>Task ID</th>
<th>Description</th>
<th>Type</th>
<th>position in TMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>task1</td>
<td>The VB 6 code should be upgraded to .NET and MSXML 6.</td>
<td>Technology changing</td>
<td>XIE itself</td>
</tr>
<tr>
<td>2</td>
<td>task2</td>
<td>The system must be able to run on a SQL 2005 DB.</td>
<td>Functional improvement</td>
<td>Structure of the Database</td>
</tr>
<tr>
<td>3</td>
<td>task3</td>
<td>All internal queues should be held in Microsoft MQ instead of IBM MQ.</td>
<td>Technological improvement</td>
<td>XIE itself</td>
</tr>
<tr>
<td>4</td>
<td>task4</td>
<td>Built in function in SQL 2005 for XML handling and optimizing should be used.</td>
<td>Technology changing</td>
<td>Structure of the Database</td>
</tr>
<tr>
<td>5</td>
<td>task5</td>
<td>The XML parsers external queues should be configured for connection many different XML sources, minimum to IBM MQ, Microsoft MQ, and XML files from share.</td>
<td>Interface improvement</td>
<td>Between SAM and TMC</td>
</tr>
<tr>
<td>6</td>
<td>task6</td>
<td>The number of connected external and internal queues should be easy to configure and maintain from the parser GUI.</td>
<td>Interface improvement and adaptability</td>
<td>XIE itself</td>
</tr>
<tr>
<td>7</td>
<td>task7</td>
<td>The error logging must be increased for better error traceability.</td>
<td>Functional improvement</td>
<td>XIE itself</td>
</tr>
<tr>
<td>8</td>
<td>task8</td>
<td>The configuration of the parser will be handled in an XML file.</td>
<td>Technology changing</td>
<td>XIE itself</td>
</tr>
<tr>
<td>9</td>
<td>task9</td>
<td>The design must be made for optimal performance of the parser, so single bundle tracking can be handled.</td>
<td>Optimization</td>
<td>XIE itself</td>
</tr>
</tbody>
</table>

**FIGURE 9. LIST OF NEW REQUIREMENT SPECIFICATIONS**
The diagram below is a network diagram showing the dependencies of the previous tasks:

M: Milestone

*FIGURE 10. NETWORK DIAGRAM AND TASK DEPENDENCIES*
3.2.2 Design Phase

The design phase consists on building an object model from the requirement specifications and the important process of reengineering that retrieve all the important functions and features of the previous version of the application.

To understand the real actions and tasks that the user can do via the new GUI, a use case diagram has been drawn because it shows exactly the interaction between the user and the application. In our case the user interaction is only done when a problem occurs during the running time of the XIE since it should work automatically in a black server room handling every minutes the coming messages from the planning system and generating others for the production system.

Also the object model is mainly described by the Class diagram that shows all the classes interacting with each others and containing all the relevant variables and functions used during the implementation of the new object oriented version of XIE.

A use of the sequence diagram is justified by the need of showing interaction between the instances of those classes and the messages exchanged between them both in the automatic mode and manual mode of running XIE.

1/ Use case diagram

This diagram describes all the actions that the user can perform using XIE.

The main actions of the applications are to Filter XML Documents either from a folder or from Queues; He can choose between either IBM Queues or MS Queues.

After filtering XML documents, the user can start parsing the XML documents and storing them in the relevant tables in the Database. He can at any time stop the parsing process and/or purge the internal queues that contain remaining XML document to be parsed.

Another main action that user can perform using XIE is to generate XML documents from XMLOutBox table in the relational Database TMC. The generated XML documents can be either stored in a folder or sent to a transmission Queue which is connected to another part of TPS system.
FIGURE 11. USE CASE DIAGRAM
2/ Class Diagram

This diagram describes all the classes obtained after the Reengineering process and needed for the new version of XIE application. As showed bellow I grouped these classes into five packages.

Package 2 contains the Data Source contains the Data Source class which is responsible for connecting the application to the Database, executing the SQL queries and managing data with Datasets. (This package will refer to Data Source layer in 3-tiers architecture model).

Package 3 and 5 contain classes’ clsMQ and clsIBM. Those classes are responsible of managing internal, reception and transmission queues in XIE application.

Package 4 contains the clsLog class which implement all the methods for creating the log File and adding all events and remarks happening during the running time of the application.

Package 1 is the main package of the XIE application and it has relations with all previous packages.
It contains Supervision class which describes the user interface and thus enables the user to perform the actions he wants.
It also contains -based on the techniques of inheritance in Object oriented design- a mother class which is connected to other classes on the other packages so that we can reuse these relations for the inherited classes.

These inherited classes are:
- Filter XML
- Parser XML
- Generate XML

FilterXML contains the methods for filtering the XML documents coming from queues or put in a folder into the relevant internal queue 1 or 2.

ParserXML contains methods for parsing XML documents from Internal queue1 or Internal queue2 into the relevant tables in the database.

GenerateXML contains methods for generating XML documents by using data available in a specific table in the database.

This package also contains an Event class handling the parsing, generating and filtering events and errors that occur during the running time of the application.

ConfigSetting is also a class that belongs to Package 1. It handles writing and reading of the configuration file so that the user can save all the changes that happen during the running time of the application.
FIGURE 12. CLASS DIAGRAM
3/ Sequence Diagram FIGURE 13

[Diagram showing sequence of events with nodes labeled XML_Supervisor and Production System, and flows indicating actions like GetMessageFromQueue() and OpenDatabaseConnection.]

These actions are performed each timer loop so each message available in the reception queue will be filtered and parsed and each data in XML, wBox will be extracted as a XML document and sent to the transmission queue.

User

Choose message source
Press start Filter
Press Start Parser1
XML message parsed()
Press Start Parser2
XML message parsed()
3.2.3 Implementation Strategy

After listing all the requirement specifications and making a new Object Oriented design with a new Object Model for the new version of XIE, the next step was to start implementing these functionalities.

At this stage of the project, an important question had to be answered: What is the best way to start implementing those functionalities in a new Object Oriented platform but taking into account the old source code implemented in VB 6.0?

To answer this question, a definition of an implementation strategy had to be made: Either implementing the new version of XIE from scratch with .NET platform and based only on the Object model, or just making a normal migration from VB 6.0 to C# and integrating the new functionalities listed above.

The final choice between those two possibilities was to implement XIE from scratch by using the Object Model and reusing some main functions from the VB6 source code (Activation/ Deactivation, Reset …etc.).

The reason of this choice is as follow:

- .NET proposes new object for parsing and generating XML document (DOM).
- New Access to the Database (ADO .NET).
- New possibilities using XML technologies available with .NET platform.
- Time reduction and reuse components.
3.2.3 Test phase

The test phase consists on the verification and validation of all methods and classes implemented in C# programming language.

First I started checking whether the new functionalities have been implemented in a good way or not and then I tested if the old actions are working better with the new implementation platform (E.g. Data Access, IBM queuing, parsing, generation and filtering of XML documents).

I also tested the GUI of the new application; Buttons, textbox, List Box…etc. have to be clear and easy to use.

Finally I tested the performance flexibility and capacity of evolution of the new version of XIE.

That was the tests made internally before deploying on site.

The deployment of the application was made via a VPN connection to the client server.
This action was very critical because our client is using the old version of XIE every day in order to transfer the production data from its production system to the production system controlling the palletizers and other machines.

The testing phase was also very critical because we should wait for their permission when their production is finished. Thus we have only three hours per day to test the application on site. We also suggest them to start running with the new version of the XIE, and if there is any problem to switch to the old version of XIE.
4 Results

The following figure represents the User Interface of the XML Integration Engine. The user can see from this interface whether the processes of filtering, parsing and generating of the XML documents are running successfully or not.

If any error occurs during the running of the application, a description of the error is displayed in the error description label and the number of the errors is displayed.

The User can check more in detailed every action fulfilled by XIE by the mean of an Error Log File. This file is created daily and contains all the information regarding the status of the connection to the database, the pinging of the reception and transmission queues and the internal processes of filtering, parsing and generation of XML documents.

In the XIE folder in Program Files, there is an XML document called App. Config file. That file contains all the setting of the application. The user can actually change the connection string of the database, change the names of the reception and transmission queues, he can also modifies the time interval for the filtering parsing and generation of the XML Documents. And also choose to run the application automatically or in manual mode, he can also change the level of details in the Log file.

In fact, the application is running in the server of the client automatically without any interaction or action from a user in the automatic mode.

The application enables the communication between the planning system which produces the distribution plan and all the orders that have to be run during a certain period of time. After sending those information using XML documents The XIE parses those information to the database of the production system and thus the palletizers and other machines on site can run Remote Mode automatically.

The XIE is now running in one client site in Baton Rouge in USA and will be deployed in other sites as soon as the clients want the new version of the XIE.
List of new improvements in the current version of XIE2006:

- Object Oriented Design
- More Evolutivity of the application
- Reuse components
- Use of Microsoft Queuing
- More compatibility with windows
- Cost reduction
- Good level of benefit from .NET technology
- Less source code
- Good access to the database
- Use of dataset
- Easy configuration
- Better error handling
- Better Integration with new systems (SPS 2006)
5 Perspectives

5.1 Theoretical perspectives

One of the challenges of software migration from legacy systems towards new object oriented design is the interaction of different programming languages and platforms because it keeps the architecture more evolutive and independent from a specific programming language.

There are several ways in which two different programming languages can interact. For example, for some languages there are mechanisms that allow calling methods or functions of one language from another. But these mechanisms are technologically challenging, not available for all languages and imply a technological overhead that can have negative effects on the software architecture and the performance of a system.[16]

Another challenge is the data exchange between the application and others, the use of new formatted messages such as XML is very interesting because it keeps the actual architecture of the system independent from its environment and thus if there is any change on a certain side of the communication channel this will not affect the other end.

Finally another theoretical perspectives for this area is to have a knowledge base of all the successful cases of software emigration in order to reuse the previous experience in similar projects.

5.2 Technical perspectives

The technical perspective of this work will focus on new features that XIE can have in the future. The following list gives an overview of what new improvements can affect XIE:

- Parsing and generation of all kinds of XML documents by adding their XML Schema
- No Software dependency (IBM queues)
- Single Bundle Tracking data stored in TMC
5 Conclusions

In this report a definition of the process of software evolution has been given. As a possible strategy of software evolution we introduced the concept of software migration especially from legacy systems towards New Object Oriented Design.

After that we try to describe some relevant phases of the migration processes taking into account the fact that the developer should start by analysing the legacy system source code and then try to build a new object oriented model for its new application.

From the theoretical background and state of the art in the area of software migration, a new framework for software migration is presented.

Finally a case study where this framework was applied for is described. The aim of this practical work was to implement a new version of XML Integration Engine. This application is used as parser of XML documents send from the production system of a company to its production system in order to store production data in the relevant tables of the central database.

The project of Software evolution is a critical process where several parameters and factors should be taken into consideration. Generally the requirement specification phase and the analysis of the source code of the previous version of the application are very determinants for the success of the project.

Also a very good understanding of the domain application of the system is needed in order to select the relevant functions from the legacy systems that have to occupy a first order in our design choices.

Most of the time, if the company hires a new or an external person to deal with the evolution or migration process in an organization find a difficulty to find a documentation and comments of the previous legacy systems because it have no UML description that can help the developer to take the relevant concerning the choice of retrieving the important functions in the source code.

The project of migration from the old version of XIE in VB6 to .NET enables me to understand theoretical aspects of the software migration process but also to suggest some new idea for migration processes especially for small and medium projects as described in the figure 9.

As a conclusion of this work We could say that the best way to deal with such projects is to take them as a real projects of software development rather than normal procedures in the life cycle of the actual application because the time expectation of such projects and also the resources needed for them should be well considered as a key factors of success.
7 References


8 Appendixes

1/ XIE Packages in Visual Studio:
2/ Folders needed during XIE running time:

Temp: Folder containing the xml filtered document before the parsing process
XML: Folder containing XML Schemas of all possible generated documents
XML_Log: Folder containing Log files daily generated
InProgress: Folder containing XML documents that have to be parsed (Manual Mode)
Processed: Folder containing a copy of the parsed and generated documents when the tasks are fulfilled successfully.
3/ App Config:

```xml
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.diagnostics>
    <!-- Logging details -->
    <add name="Triggers" value="0" />
    <add name="Application" value="3" />
    <add name="Database" value="0" />
    <add name="TreeView" value="4" />
    <add name="MethodDetails" value="0" />
  </switches>
  <appSettings>
    <add key="GUI_Type" value="1" />
    <add key="AutoStart" value="1" />
    <add key="FromQueue" value="no" />
    <add key="FromFolder" value="yes" />
    <add key="useConsoleLog" value="true" />
    <add key="keepParsedFiles" value="true" />
    <add key="strConnectionString" value="Data Source=TMC3;Initial Catalog=Advocate;User ID=sa; Password=vfr6710" />
    <add key="QueueType" value="IBM" />
    <add key="QueueType" value="Microsoft" />
    <add key="strIBM_RecQueueID" value="TO.TPS.DATA" />
    <add key="strIBM_RecQueueManagerID" value="HRTTPS" />
    <add key="strSourcePath" value="C:\schur\InProgress" />
    <add key="tmrLoopInterval_Filter" value="1000" />
    <add key="strDBConnection_Filter" value="N/A" />
    <add key="strReception_Path" value=".\private\reception_queue" />
    <add key="strInternal_1_Path" value=".\private\internal1_queue" />
    <add key="strInternal_2_Path" value=".\private\internal2_queue" />
    <add key="ErrorMessage" value="N/A" />
    <add key="ErrorNumber" value="0" />
    <add key="SourceError" value="N/A" />
    <add key="StatusGeneration" value="--" />
    <add key="StatusParser1" value="--" />
    <add key="StatusParser2" value="--" />
    <add key="GeneratorDBConnection" value="N/A" />
    <add key="GeneratorLoopInterval" value="3000" />
    <add key="SupervisionLoopInterval" value="2000" />
    <add key="strTransmissionQueueID" value="TO.TPS.DATA" />
    <add key="strTransmissionManagerID" value="HRTTPS" />
    <add key="PingReqStatus" value="--" />
    <add key="PingResStatus" value="--" />
    <add key="CtrlTransmissQueueID" value="TO.SAM.CONTROL" />
    <add key="CtrlTransmissionManagerID" value="HRTTPS" />
    <add key="ReceptionQueueID" value="TO.TPS.DATA" />
    <add key="ReceptionManagerID" value="HRTTPS" />
    <add key="Destination_path" value="c:\schur\Processed\Generated\" />
    <add key="useHelpTexts" value="true" />
  </appSettings>
</system.diagnostics>
</configuration>
```
4 / Example of an XML Parsed Document:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<DistributionPlan Dateline="2005-04-24" PartnerID="TPS" ResponseMode="Initial" Shift="WED1">
  <Products Dateline="2005-04-24" PartnerID="TPS" Shift="WED1">
    <Product BundleSize="50" Description="COMIC" ID="2302092">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>50</MaxBundleSize> <MinBundleSize>50</MinBundleSize>
      <ProductCode>COMIC05</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="340" Description="COMIC" ID="2302091">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>340</MaxBundleSize> <MinBundleSize>340</MinBundleSize>
      <ProductCode>COMIC24</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="60" Description="COMIC STATE" ID="2302122">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>60</MaxBundleSize> <MinBundleSize>60</MinBundleSize>
      <ProductCode>COMST6</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="40" Description="COMIC STATE" ID="2302121">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>40</MaxBundleSize> <MinBundleSize>40</MinBundleSize>
      <ProductCode>COMST3</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="60" Description="COMIC STATE" ID="2302119">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>60</MaxBundleSize> <MinBundleSize>60</MinBundleSize>
      <ProductCode>COMST4</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="60" Description="COMIC STATE" ID="2302118">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>60</MaxBundleSize> <MinBundleSize>60</MinBundleSize>
      <ProductCode>COMST5</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="70" Description="COMIC STATE" ID="2302117">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>70</MaxBundleSize> <MinBundleSize>70</MinBundleSize>
      <ProductCode>COMST7</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="60" Description="COMIC STATE" ID="2302116">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>60</MaxBundleSize> <MinBundleSize>60</MinBundleSize>
      <ProductCode>COMST2</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="340" Description="COMIC STATE" ID="2302115">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>340</MaxBundleSize> <MinBundleSize>340</MinBundleSize>
      <ProductCode>COMST8</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="50" Description="COMIC" ID="2302114">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>50</MaxBundleSize> <MinBundleSize>50</MinBundleSize>
      <ProductCode>COMIC07</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="50" Description="COMIC" ID="2302113">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>50</MaxBundleSize> <MinBundleSize>50</MinBundleSize>
      <ProductCode>COMIC13</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
    <Product BundleSize="340" Description="COMPLETE COMIC" ID="2365237">
      <IssueDate>2005-04-24</IssueDate> <MaxBundleSize>340</MaxBundleSize> <MinBundleSize>340</MinBundleSize>
      <ProductCode>CMPCM1</ProductCode> <PubID>243757</PubID> <RecircLimit>0</RecircLimit>
    </Product>
  </Products>
</DistributionPlan>
```
<Product BundleSize="50" Description="COMIC" ID="2302110"
  IssueDate="2005-04-24" MaxBundleSize="50" MinBundleSize="50"
  ProductCode="COMIC06" PubID="243757" RecircLimit="0" />
  <Product BundleSize="50" Description="COMIC" ID="2302109"
  IssueDate="2005-04-24" MaxBundleSize="50" MinBundleSize="50"
  ProductCode="COMIC04" PubID="243757" RecircLimit="0" />
  <Product BundleSize="50" Description="COMIC" ID="2302108"
  IssueDate="2005-04-24" MaxBundleSize="50" MinBundleSize="50"
  ProductCode="COMIC14" PubID="243757" RecircLimit="0" />
  <Product BundleSize="50" Description="COMIC" ID="2302107"
  IssueDate="2005-04-24" MaxBundleSize="100" MinBundleSize="100"
  ProductCode="COMIC12" PubID="243757" RecircLimit="0" />
  <Product BundleSize="50" Description="COMIC" ID="2302106"
  IssueDate="2005-04-24" MaxBundleSize="50" MinBundleSize="50"
  ProductCode="COMIC16" PubID="243757" RecircLimit="0" />
  <Product BundleSize="50" Description="COMIC" ID="2302105"
  IssueDate="2005-04-24" MaxBundleSize="60" MinBundleSize="60"
  ProductCode="COMIC19" PubID="243757" RecircLimit="0" />
  <Trucks Dateline="2005-04-24" PartnerID="TPS" Shift="WED1">
    <Bundles>
      <Bundle Barcode="457777" Batches="1" CombinedAccounts="false" Count="673"
        ID="2457777" InfoSheetLayout="0" Pages="92" Size="50"
        StartingSequence="1" TopSheetLayout="6" Type="Standard" />
      <Bundle Barcode="457778" Batches="1" CombinedAccounts="false" Count="1"
        ID="2457778" InfoSheetLayout="0" Pages="92" Size="42"
        StartingSequence="674" TopSheetLayout="7" Type="Key" />
    </Bundles>
    <Run BundlesPerContainer="55" Draw="19514" ID="2379932"
      IssueDate="2005-04-24" MultiRunID="2459030" ProductCode="CMPCM1"
      ProductID="2365237" PubID="243757" RunCode="STORA-CMPCM1"
      Target="Storage" />
    <Bundles>
      <Bundle Barcode="459033" Batches="1" CombinedAccounts="false" Count="57"
        ID="2459033" InfoSheetLayout="0" Pages="8" Size="340"
        StartingSequence="1" TopSheetLayout="6" Type="Standard" />
      <Bundle Barcode="459034" Batches="1" CombinedAccounts="false" Count="1"
        ID="2459034" InfoSheetLayout="0" Pages="8" Size="134"
        StartingSequence="58" TopSheetLayout="7" Type="Key" />
    </Bundles>
    <Loader ID="P3" Role="Included" />
    <Run>
      <Run BundlesPerContainer="66" Draw="36137" ID="2379912"
        IssueDate="2005-04-24" MultiRunID="2457793" ProductCode="COMIC09"
        ProductID="2302100" PubID="243757" RunCode="STORA-COMIC09"
        Target="Storage" />
      <Bundles>
        <Loader ID="P3" Role="Included" />
        <Run>
          <Run BundlesPerContainer="66" Draw="36137" ID="2379912"
            IssueDate="2005-04-24" MultiRunID="2457793" ProductCode="COMIC09"
            ProductID="2302100" PubID="243757" RunCode="STORA-COMIC09"
            Target="Storage" />
          <Bundles>
<Bundle Barcode="457796" Batches="1" CombinedAccounts="false" Count="722"
  ID="457796" InfoSheetLayout="0" Pages="108" Size="50"
  StartingSequence="1" TopSheetLayout="6" Type="Standard" />
</Bundle>

<Bundle Barcode="457797" Batches="1" CombinedAccounts="false" Count="1"
  ID="457797" InfoSheetLayout="0" Pages="108" Size="37"
  StartingSequence="723" TopSheetLayout="7" Type="Key" />
</Bundle>

<Loaders>
  <Loader ID="P3" Role="Included" />
</Loaders>
</Run>

<Run BundlesPerContainer="66" Draw="32932" ID="2379906"
  IssueDate="2005-04-24" MultiRunID="2457765" ProductCode="COMIC02"
  ProductID="2302094" PubID="243757" RunCode="STORA-COMIC02"
  Target="Storage">
  <Bundles>
    <Bundle Barcode="457768" Batches="1" CombinedAccounts="false" Count="658"
      ID="457768" InfoSheetLayout="0" Pages="106" Size="50"
      StartingSequence="1" TopSheetLayout="6" Type="Standard" />
  </Bundles>
</Run>

<Loaders>
  <Loader ID="P4" Role="Included" />
</Loaders>
</Run>

<Run BundlesPerContainer="1" Draw="75" ID="2379911" IssueDate="2005-04-24"
  MultiRunID="2379786" ProductCode="COMIC23" ProductID="2302099"
  PubID="243757" RunCode="STORA-COMIC23" Target="Storage">
  <Bundles>
    <Bundle Barcode="379952" Batches="1" CombinedAccounts="false" Count="1"
      ID="379952" InfoSheetLayout="0" Pages="42" Size="75"
      StartingSequence="1" TopSheetLayout="7" Type="Key" />
  </Bundles>
</Run>

<Loaders>
  <Loader ID="P4" Role="Included" />
</Loaders>
</Run>
</Truck>
</Trucks>
</DistributionPlan>
5/ Structure of the tables in the database

XML Integration Engine is based on a set of tables as described in the following section.

Metadata
Valid Elements are defined within [XMLElements] and corresponding attributes within [XMLAttributes] as depicted in Figur 1.

Figur 1. Definition of XML Elements and Attributes.

Each column is described in sections 0 and 0

Table: XMLElements

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Varchar</td>
<td>Element Name</td>
</tr>
<tr>
<td>Description</td>
<td>Varchar</td>
<td></td>
</tr>
</tbody>
</table>

Table: XMLAttributes

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Varchar</td>
<td>Attribute Name</td>
</tr>
<tr>
<td>ColumnID</td>
<td>Varchar</td>
<td>Used for mapping information in the TMC Database Interface Tables (XMLInBox and XMLOutBox) to XML Element Attributes. ColumnID holds the Names of the columns in the Interface Tables.</td>
</tr>
<tr>
<td>ElementRef</td>
<td>Varchar</td>
<td>Reference to XML Element holding this attribute.</td>
</tr>
<tr>
<td>Sequence</td>
<td>Int</td>
<td>Used to insert Attribute Values supplied with XML Documents into the TMC Database Inbox in the correct order – disregard of the order of the supplied attributes.</td>
</tr>
</tbody>
</table>
**In- / Out Boxes**

On successful parsing of XML Documents by the XIE, the Element- and Attribute Values are stored within \[XMLInBox1\] / \[XMLInBox1\] in the TMC Database and subsequently handled by the Stored Procedures listed below.

Data from the TMC Database to SAM is stored within \[XMLOutBox\]. From here it is read by the XIE and subsequently processed into an XML Document.

Figur 2. TMC In- and Out Boxes.
### Table: XMLInBox1/2 (Where data are parsed to)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemID</td>
<td>Int</td>
<td>Unique Id within XMLInBox for this element.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Int</td>
<td>ItemID of Parent Element.</td>
</tr>
<tr>
<td>MessageID</td>
<td>Varchar</td>
<td>MessageID identifying message type. Ex.: 100404 (DistributionPlan)</td>
</tr>
<tr>
<td>ElementName</td>
<td>Varchar</td>
<td>Valid XML Element.</td>
</tr>
<tr>
<td>Col1</td>
<td>Varchar</td>
<td>Element Value – if supplied.</td>
</tr>
<tr>
<td>Col2 – Col15</td>
<td>Varchar</td>
<td>Attribute Values for current element.</td>
</tr>
<tr>
<td>Status</td>
<td>Varchar</td>
<td><strong>Ready:</strong> Message ready for processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Processed:</strong> Message processed by Parser.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Error:</strong> Error processing message.</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Varchar</td>
<td>Timestamp when message generated.</td>
</tr>
<tr>
<td>InsertedBy</td>
<td>Varchar</td>
<td>ID of Application (TPS)</td>
</tr>
<tr>
<td>MessageGroup</td>
<td>Varchar</td>
<td>ID uniquely grouping elements belonging to a message.</td>
</tr>
</tbody>
</table>

### Table: XMLOutBox (Where data are generated from)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemID</td>
<td>Int</td>
<td>Unique Id within XMLOutBox for this element.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Int</td>
<td>ItemID of Parent Element.</td>
</tr>
<tr>
<td>MessageID</td>
<td>Varchar</td>
<td>MessageID identifying message type. Ex.: 100010 (RequestProductionData)</td>
</tr>
<tr>
<td>ElementName</td>
<td>Varchar</td>
<td>Valid XML Element.</td>
</tr>
<tr>
<td>Col1</td>
<td>Varchar</td>
<td>Element Value – if supplied.</td>
</tr>
<tr>
<td>Col2 – Col15</td>
<td>Varchar</td>
<td>Attribute Values for current element.</td>
</tr>
<tr>
<td>Status</td>
<td>Varchar</td>
<td><strong>Ready:</strong> Message ready for processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Transferred:</strong> Message processed by Generator</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Varchar</td>
<td>Timestamp when message generated.</td>
</tr>
<tr>
<td>InsertedBy</td>
<td>Varchar</td>
<td>ID of Application (TPS)</td>
</tr>
<tr>
<td>Origin</td>
<td>Varchar</td>
<td>ID of Origin (TPS). Send as part of message to SAM.</td>
</tr>
<tr>
<td>Persistence</td>
<td>Int</td>
<td>0: Message does not survive shut down of MQ Server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Message survives shut down of MQ Server.</td>
</tr>
</tbody>
</table>
6/ Stored procedures

The procedures generated to support the transfer of data between SAM and TMC are listed below.

<table>
<thead>
<tr>
<th>Stored Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp_AddEdition</td>
<td>Adds Products to [Edition]</td>
</tr>
<tr>
<td>sp_AddOrder</td>
<td>Adds Run to [Order]</td>
</tr>
<tr>
<td>sp_AssignRouteToEdition</td>
<td>Assigns Routes to Editions within [Route Assignment] on addition of Run.</td>
</tr>
<tr>
<td>Sp_GetDLPNo</td>
<td>Returns DLPNo for current Truck (Route). If not found it is created within DLP</td>
</tr>
<tr>
<td>sp_InsertIntoXMLInBox1</td>
<td>Called by VB to Insert Data to [XMLInBox$]</td>
</tr>
<tr>
<td>sp_InsertIntoXMLInBox2</td>
<td></td>
</tr>
<tr>
<td>sp_ReportBundlesDelivered</td>
<td>Triggered on update of OrderStatus to 70 (Completed) within [Order] or update of ShiftStatus to 70 (Completed) within [Shift] Generates Report “Bundles Delivered”</td>
</tr>
<tr>
<td>sp_RequestProductionData</td>
<td>Called by TMC to request Production Data from SAM</td>
</tr>
<tr>
<td>sp_ResequenceOrders</td>
<td>Called on reception of Run with Action = “Resequence”. Resequences orders by priority.</td>
</tr>
<tr>
<td>sp_ResequenceTrucks</td>
<td>Called on reception of Truck with Action = “Resequence”. Resequences Trucks by priority.</td>
</tr>
<tr>
<td>sp_TransferFromXMLInBox1</td>
<td>Called on reception of ProductionPlan Message from SAM.</td>
</tr>
<tr>
<td>sp_TransferFromXMLInBox2</td>
<td></td>
</tr>
<tr>
<td>sp_TransferProductsFromXMLInBox1</td>
<td>Called on reception of Products Message from SAM. Inserts Product Information to [Edition]</td>
</tr>
<tr>
<td>sp_TransferProductsFromXMLInBox2</td>
<td></td>
</tr>
<tr>
<td>sp_TransferTieLineAssignFromXMLInBox1</td>
<td>Called on reception of TieLineAssign Message from SAM.</td>
</tr>
<tr>
<td>sp_TransferTieLineAssignFromXMLInBox2</td>
<td></td>
</tr>
<tr>
<td>sp_TransferTrucksFromXMLInBox1</td>
<td>Called on reception of Trucks Message from SAM. Updates or inserts Truck Information to [Route]</td>
</tr>
<tr>
<td>sp_TransferTrucksFromXMLInBox2</td>
<td></td>
</tr>
<tr>
<td>sp_UpdateEdition</td>
<td>Called on reception of Product. Updates the specified Product within [Edition]</td>
</tr>
<tr>
<td>sp_UpdateOrder</td>
<td>Called on reception of Run. Updates the specified Run within [Order]</td>
</tr>
<tr>
<td>sp_UpdatePrioritiesBeforeNewRun</td>
<td>Called prior to insertion of new Run into [Order]. Increments priorities on Orders before which the new Run is inserted.</td>
</tr>
<tr>
<td>sp_UpdatePrioritiesBeforeNewTruck</td>
<td>Called prior to insertion of new Runs as supplied with Truck. Increments priorities on Orders before which the new Runs are inserted.</td>
</tr>
<tr>
<td>sp_UpdateXMLInBoxRunElements1</td>
<td>Updates Status on processed records within [XMLInBox$] to ‘Processed’</td>
</tr>
<tr>
<td>sp_UpdateXMLInBoxRunElements2</td>
<td></td>
</tr>
</tbody>
</table>
XML Schema

This page illustrates the general Schema definition used actually for generating XML documents. This representation can contain as many attributes as we desire just by adding new XML schema on the folder C:schurXML.