

## **CHAPTER 2**

### **THEORETICAL FOUNDATION**

#### **2.1 Theoretical Foundation**

In order to understand this project better, this thesis need to give several theoretical foundation, which will be the basic theory that this thesis will be applied for complete the Human Resource application software. Those theoretical foundations that will be used as the basic theory for this thesis are:

##### **2.1.1 Application Software**

Computer program is made by human to fulfill the user needs. Computer program consists of some applications that created by human based on the user requirements. Application software is made to generate the manual process into more automatic process in order to make the user's work easier and faster. That is the reason of why we should make the application as user friendly as possible.

Basically, application can be divided into two categories, which are system software and application software.

##### **1. System software**

System software is different with the application software because it consists of programs from the low level in order to communicate with

the computer at the most basic level. That is the reason of people to call system software as the “low - level software” [2].

According to [3], it is said that: “System software refers to the file and programs that make up your computer’s operating system. The programs that are the part of the system software include assemblers, compilers, file management tools, system utilities, and debuggers.

## 2. Application software

Application software is the one that people use, rather than system software, because it is high - level software, so it is easier to understand. People usually called this application software as the “end – user program”. The programs that are the part of the application software include the database program, spreadsheet, word processing. However, the application software cannot run without system software, because application software has the position on top of the system software. That is why the system software is also important because it supports and become the basic for the application software [2].

There is also a definition about the application software that comes from [4].

The definition of application software according to them is:

“Application software is a subclass of computer software that employs the capabilities of a computer directly and thoroughly to a task that the user wishes to perform”.

### **2.1.2 Human Resource Management**

Human Resource Management (HRM) is the process of acquiring, training, appraising, and compensating employees, and attending to their labor relations, health and safety, and fairness concerns. Human resource involved in carrying out the “people” or human resource aspects of a management position, including recruiting, screening, training, rewarding, and appraising. Its main goal is to enable employees to work at their maximum level of efficiency [5].

Human resource management in general can be divided into several jobs, which can be applied in an organization. Those jobs are:

- **Planning**

Planning includes the activities of establishing goals and standards, developing rules and procedures, developing plans and forecasting.

- **Organizing**

Organizing includes the activities of giving each subordinate a specific task, establishing departments, delegating authority to subordinates, establishing channels of authority and communication, coordinating the work of subordinates.

- **Staffing**

Staffing includes the activities of determining what type of people should be hired, recruiting prospective employees, selecting employees, setting performance standards,

compensating employees, evaluating performance, counseling employees, training and developing employees.

- Leading

Leading includes the activities of getting others to get the job done, maintaining morale, motivating subordinates.

- Controlling


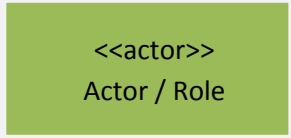

Controlling includes the activities of setting standards such as sales quotas, quality standards, or production levels, checking to see how actual performance compares with these standards, taking corrective action as needed [6].

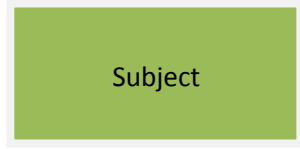

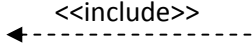
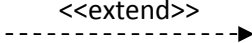
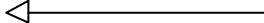
### **2.1.3 UML**

UML or Unified Modeling Language was released by Object Management Group (OMG). UML is a unifying language, which created with the purpose to help IT professionals in order to make the model of computer application. UML has become a standard modeling language because UML is programming – language independent. UML has several standard diagrams, such as use case diagram, class diagram, sequence diagram, state chart diagram, activity diagram, component diagram, and deployment diagram. This thesis will use several UML diagrams, like use case diagram, class diagram, sequence diagram, and activity diagram, which will become the model of HR application [7].

### 2.1.4 Use Case Diagram

Use Case Diagram is a functional diagram in that it portrays the basic function of the system, which is to know what the users can do and how the system should respond to the user's action. These are several definitions and syntax of the symbols inside a use case diagram [8]:

No.	DEFINITION	SYMBOL
1.	<b>An Actor:</b> <ul style="list-style-type: none"> <li>Is a person or system that derives benefit from and is external to the subject.</li> <li>Is depicted as either a stick figure (default) or if a nonhuman actor is involved, as a rectangle with &lt;&lt;actor&gt;&gt; in it (alternative).</li> <li>Is labeled with its role.</li> <li>Can be associated with other actors using a specialization / superclass association, denoted by an arrow with a hollow arrowhead.</li> <li>Is placed outside the subject boundary.</li> </ul>	 <p>Actor / Role</p> 
2.	<b>A Use Case:</b> <ul style="list-style-type: none"> <li>Represents a major piece of system functionality.</li> <li>Can extend another use case.</li> <li>Can include another use case.</li> <li>Is placed inside the system boundary.</li> <li>Is labeled with a descriptive verb-noun phrase.</li> </ul>	

3.	<b>A Subject Boundary:</b> <ul style="list-style-type: none"> <li>Includes the name of the subject inside or on top.</li> <li>Represents the scope of the subject, for example a system or an individual business process.</li> </ul>	
4.	<b>An Association Relationship:</b> <ul style="list-style-type: none"> <li>Links an actor with the use case(s) with which it interacts.</li> </ul>	
5.	<b>An Include Relationship:</b> <ul style="list-style-type: none"> <li>Represents the inclusion of the functionality of one use case within another.</li> <li>It has an arrow drawn from the base use case to the used use case.</li> </ul>	
6.	<b>An Extend Relationship:</b> <ul style="list-style-type: none"> <li>Represents the extension of the use case to include optional behavior.</li> <li>It has an arrow drawn from the extension use case to the base use case.</li> </ul>	
7.	<b>A Generalization Relationship:</b> <ul style="list-style-type: none"> <li>Represents a specialized use case to a more generalized one.</li> <li>It has an arrow drawn from the specialized use case to the base use case.</li> </ul>	

**Table 1 Use Case Diagram Definitions and Syntaxes**

### **2.1.5 Use Case Description**

Use Case Description is a tool that contains all the information needed to build the diagrams that follow, but it expresses the information in a less formal way that is usually simpler for users to understand. Use Case description usually has three basic parts, which are:

- Overview information

Overview information identifies the use case and provides basic background information about the use case.

- Relationship

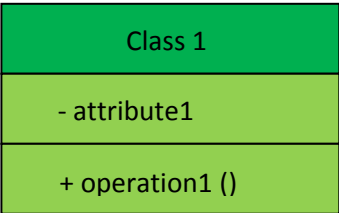
Relationship explains how the use case is related to other use cases and users.

- The flow of events

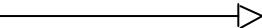


Flow of events describes the individual steps within the business process [8].

### **2.1.6 Class Diagram**

Class Diagram is a static model that shows the classes and the relationships among classes that remain constant in the system over time. It depicts classes, which include both behaviors a state, with the relationships between the classes. These are several definitions and syntax of the symbols inside a class diagram [8]:

No.	DEFINITION	SYMBOL
1.	<p><b>A Class:</b></p> <ul style="list-style-type: none"> <li>Represents a kind of person, place, or thing about which the system will need to capture and store information.</li> <li>It has a name typed in bold and centered in its top compartment.</li> <li>It has a list of attributes in its middle compartment.</li> <li>It has a list of operations in its bottom compartment.</li> <li>It does not explicitly show operations that are available to all classes.</li> </ul>	
2.	<p><b>An Attributes:</b></p> <ul style="list-style-type: none"> <li>Represent properties that describe the state of an object.</li> <li>Can be derived from other attributes, shown by placing a slash before the attribute's name.</li> </ul>	<p>Attribute name</p> <p>/derived attribute name</p>
3.	<p><b>An Operation:</b></p> <ul style="list-style-type: none"> <li>Represents the actions or functions that a class can perform.</li> <li>Can be classified as a constructor, query, or update operation.</li> <li>Includes parentheses that may contain parameters or information needed to perform the operation.</li> </ul>	<p>operation name ()</p>


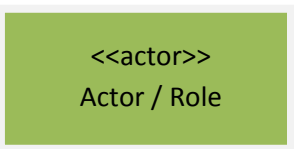
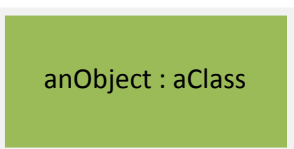




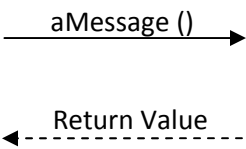
4.	<b>An Association:</b> <ul style="list-style-type: none"> <li>Represents a relationship between multiple classes or a class and itself.</li> <li>It is labeled using a verb phrase or a role name, whichever better represents the relationship.</li> <li>Can exist between one or more classes.</li> <li>Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance.</li> </ul>	<p style="text-align: center;">AssociatedWith</p> <hr style="width: 50%; margin: auto;"/> <p style="text-align: center;">0..* <span style="float: right;">1</span></p>
5.	<b>A Generalization:</b> <ul style="list-style-type: none"> <li>Represents a kind-of-relationship between multiple classes.</li> </ul>	
6.	<b>An Aggregation:</b> <ul style="list-style-type: none"> <li>Represents a logical a-part-of relationship between multiple classes or a class and itself.</li> <li>It is a special form of an association.</li> </ul>	<p>0..* IsPartOf ► 1 </p>
7.	<b>A Composition:</b> <ul style="list-style-type: none"> <li>Represents a physical a-part-of relationship between multiple classes or a class and itself.</li> <li>It is a special form of an association.</li> </ul>	<p>1..* IsPartOf ► 1 </p>

**Table 2 Class Diagram Definitions and Syntaxes**

### 2.1.7 Sequence Diagram

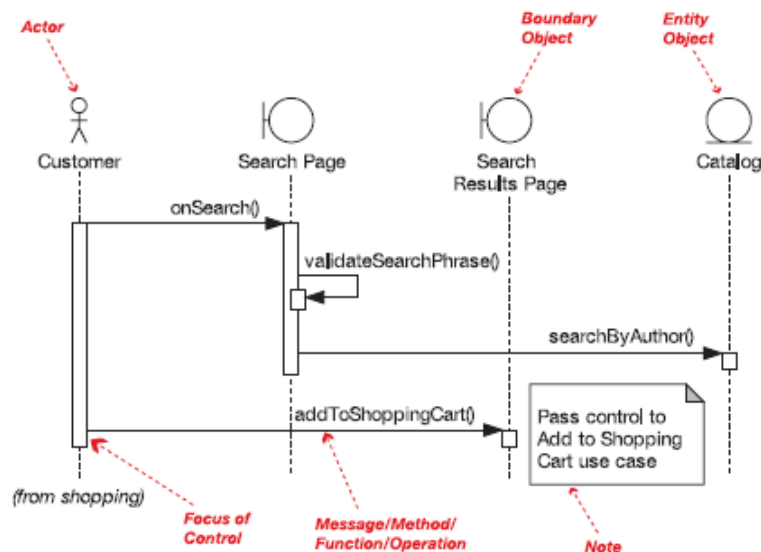
Sequence Diagram is one of two types of interaction diagrams. It is a dynamic model that shows the explicit sequence of messages that are passed between objects in a defined interaction. These are several definitions and syntax of the symbols inside a class diagram [8]:

No.	DEFINITION	SYMBOL
1.	<b>An actor:</b> <ul style="list-style-type: none"> <li>It is a person or system that derives benefit from and is external to the system.</li> <li>Participates in a sequence by sending and/or receiving messages.</li> <li>It is placed across the top of the diagram.</li> <li>It is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with &lt;&lt;actor&gt;&gt; in it (alternative).</li> </ul>	 Actor / Role 
2.	<b>An Object:</b> <ul style="list-style-type: none"> <li>Participates in a sequence by sending and/or receiving messages.</li> <li>It is placed across the top of the diagram.</li> </ul>	
3.	<b>A Lifeline:</b> <ul style="list-style-type: none"> <li>Denotes the life of an object during a sequence.</li> <li>Contains an X at the point at which the class no longer interacts.</li> </ul>	

4.	<b>An Execution Occurrence:</b> <ul style="list-style-type: none"> <li>It is a long narrow rectangle placed a top a lifeline.</li> <li>Denotes when an object is sending or receiving messages.</li> </ul>	
5.	<b>A Message:</b> <ul style="list-style-type: none"> <li>Conveys information from one object to another one.</li> <li>An operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.</li> </ul>	

**Table 3 Sequence Diagram Definitions and Syntaxes**

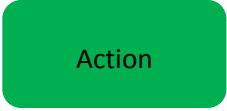
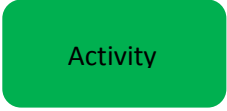
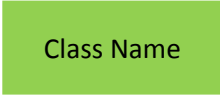
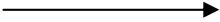
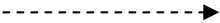



There are other syntaxes used in sequence diagram [9]:

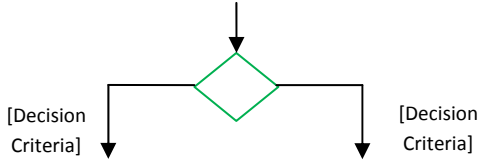
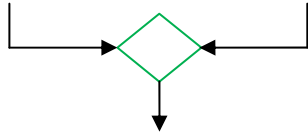


**Figure 1**  
**Sequence Diagram**  
**Other Syntaxes**

### 2.1.8 Activity Diagram

Activity Diagram is used to model the behavior in a business process independent of objects. It portrays the primary activities and the relationships among the activities in a process. These are several definitions and syntax of the symbols inside an activity diagram [8]:

No.	DEFINITION	SYMBOL
1.	<b>An Action:</b> <ul style="list-style-type: none"> <li>It is a simple, no decomposable piece of behavior.</li> <li>It is labeled by its name.</li> </ul>	
2.	<b>An Activity:</b> <ul style="list-style-type: none"> <li>It is used to represent a set of actions.</li> <li>It is labeled by its name.</li> </ul>	
3.	<b>An Object Node:</b> <ul style="list-style-type: none"> <li>It is used to represent an object that is connected to a set of object flows.</li> <li>It is labeled by its class name.</li> </ul>	
4.	<b>A Control Flow:</b> <ul style="list-style-type: none"> <li>Shows the sequence of execution.</li> </ul>	
5.	<b>An Object Flow:</b> <ul style="list-style-type: none"> <li>Shows the flow of an object from one activity (or action) to another activity (or action).</li> </ul>	
6.	<b>An Initial Node:</b> <ul style="list-style-type: none"> <li>Portrays the beginning of a set of actions or activities.</li> </ul>	
7.	<b>A Final – Activity Node:</b> <ul style="list-style-type: none"> <li>It is used to stop all flows and object flows in an activity (or action).</li> </ul>	
8.	<b>A Final – Flow Node:</b> <ul style="list-style-type: none"> <li>It is used to stop a specific control flow or object flow.</li> </ul>	

9.	<b>A Decision Node:</b> <ul style="list-style-type: none"> <li>• It is used to represent a test condition to ensure that the control flow or object flow only goes down one path.</li> <li>• It is labeled with the decision criteria to continue down the specific path.</li> </ul>	
10.	<b>A Merge Node:</b> <ul style="list-style-type: none"> <li>• It is used to bring back together different decision paths that were created using a decision node.</li> </ul>	

**Table 4 Activity Diagram Definitions and Syntaxes**

### 2.1.9 ASP.NET

ASP.NET was created by Microsoft. ASP.NET is a free technology that can be used by programmer to create the dynamic web applications. By using ASP.NET, programmer is able to create any application from small range, personal website, until enterprise – class application [10].

ASP.NET is a unified model of web development that has the purpose to help programmer in build the enterprise – class web application by using the minimum number of coding. ASP.NET contains several functions, including [11]:

- A Page and controls framework
- The ASP.NET compiler
- Security infrastructure
- State-management facilities

- Application configuration
- Health monitoring and performance features
- Debugging support
- An XML Web services framework
- Extensible hosting environment and application life cycle management

There several steps to do to connect ASP.NET with database: create a database connection, create a database command, create a datareader, bind to a repeater control, and close the database connection [12].

#### **2.1.10 Database**

There are several terms in database, which should be known as the basic of database to the application [13]:

##### **2.1.10.1 Database Definition**

Database is a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization.

##### **2.1.10.2 Data Manipulation**

There are some SQL Data Manipulation Language statements that usually be used by people, which are:

- SELECT

Select is used to query data in the database. The purpose to use Select statement is to retrieve and display data from

one or more database tables. It is the most frequently used SQL command, which has the general form as follow:

```
SELECT      [DISTINCT | ALL] { * | [columnExpression
               [AS newName]] [...]}
FROM        TableName [alias] [...]
[WHERE       condition]
[GROUP BY   columnList] [HAVING condition]
[ORDER BY   columnList]
```

- **INSERT**

Insert is used to insert data into a table. The general form of Insert statement is:

```
INSERT INTO TableName [(columnList)]
VALUES (data ValueList)
```

- **UPDATE**

Update is used to update data in a table. It allows the contents of existing rows in a named table to be changed.

The general form of Update statement is:

```
UPDATE TableName
SET columnName1 = dataValue1 [,columnName2 =
    dataValue2...]
[WHERE searchCondition]
```

- DELETE

Delete is used to delete data from a table. It allows rows to be deleted from a named table. The general form of Delete statement is:

<b>DELETE FROM</b> TableName [ <b>WHERE</b> searchCondition]
---

### 2.1.10.3 Normalization

Normalization is a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise. The purpose of normalization is to identify a suitable set of relations that support the data requirements of an enterprise. This suitable set of relations has several characteristics, including:

- The *minimal* number of attributes necessary to support the data requirements of the enterprise.
- Attributes with a close logical relationship (described as functional dependency) are found in the same relation.
- *Minimal* redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys, which are essential for the joining of related relations.



Relational databases also rely on the existence of a certain amount of data redundancy. This redundancy is in the form of copies of primary keys (or candidate keys) acting as foreign keys in related relations to enable the modeling of relationships between data.

### **2.1.11 Entity – Relationship Diagram**

There are several terms in entity – relationship diagram [14]:

#### **2.1.11.1 Entities**

Entities are any data in the user environment, which want to be maintained inside the organization. These data can be in the form of person, place, object, event, or event a concept.

The examples of entities:

- Person: STUDENT, LECTURER, EMPLOYEE
- Place: COUNTRY, STATE, CITY
- Object: COMPUTER, BALL, BUILDING
- Event: REGISTRATION, LOGIN, SEARCH
- Concept: ACCOUNT, COURSE, WORK CENTER

#### **2.1.11.2 Attributes**

Attributes are the characteristic of each entity, which become the interest of the organization.

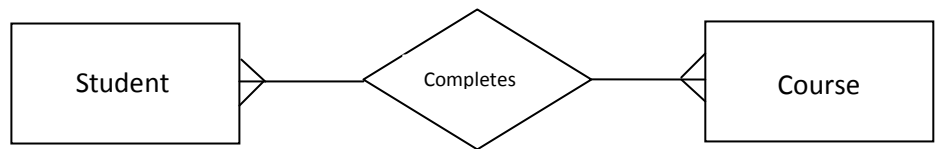
The examples of attributes:

- STUDENT: student\_number, student\_name, student\_phone

- COUNTRY: country\_ID, country\_name, country\_place
- COMPUTER: computer\_no, computer\_tool, computer\_color

### 2.1.11.3 Relationship

Relationship is the association between the instances of one or more entity types, which become the interest of the organization. For example, there is a course that should be taken by each of the student in a university. The university wants to know which Courses each of its Students has completed. This interest leads to a relationship between Student and Course, which called Completes.

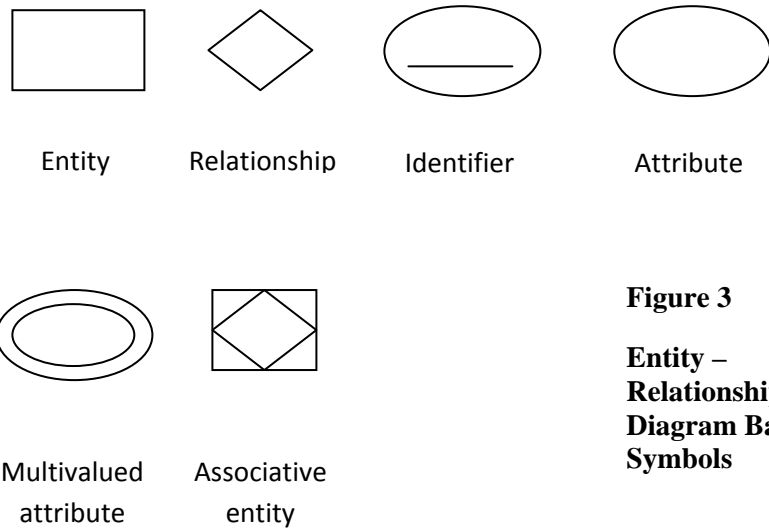


**Figure 2 Database Relationship**

### 2.1.11.4 Entity – Relationship Diagram

Entity – Relationship Diagram (ERD) is a detailed, logical, and graphical representation of the data in an organization or business area, which contains the relationship or associations among the entities, and the attributes or properties of both the entities and their relationship. There are also several symbols that will be used in entity - relationship diagram:

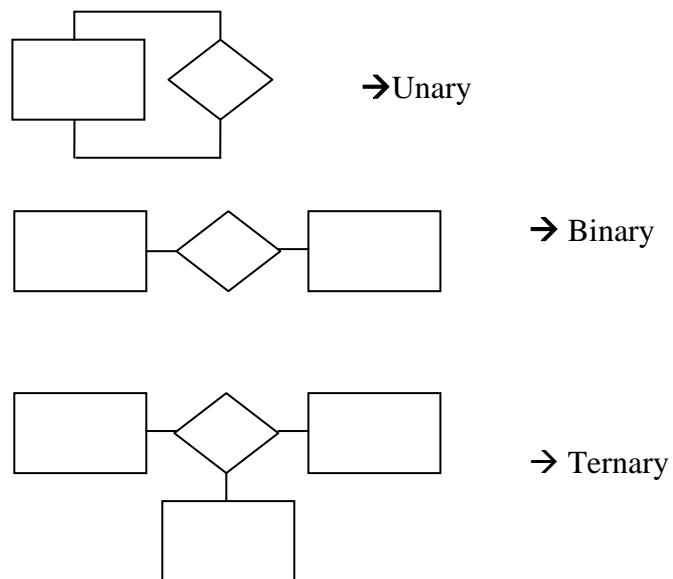
- Basic symbols:



**Figure 3**

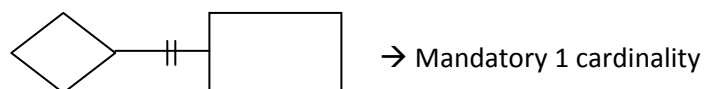
**Entity –  
Relationship  
Diagram Basic  
Symbols**

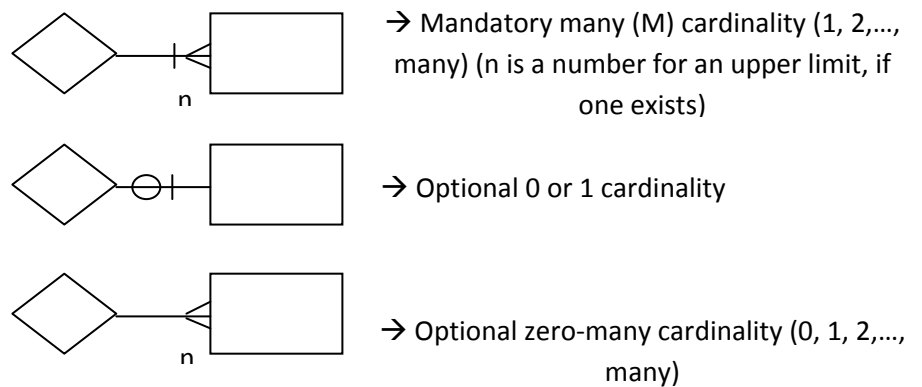
- Relationship degree:



**Figure 4 Entity – Relationship Diagram Relationship Degree**

- Relationship cardinality:





**Figure 5 Entity – Relationship Diagram Relationship Cardinality**

### 2.1.12 Microsoft SQL Server 2005

Microsoft SQL Server is a tool for Database Management System (DBMS), which is created by Microsoft. It is a computer application used to create desktop, enterprise, and web-based database systems. It can be used at different levels and with various goals [15].

Several benefits that we can get when we use Microsoft SQL Server:

- Microsoft SQL Server is able to work as a natural extension of Windows 2000, Windows Server 2003, and Windows XP. The reason for this is because Microsoft SQL Server is really closely integrated with those operating systems. By having this benefit, user does not have to learn another user interface in order for them to work with this database system.
- Microsoft SQL Server has a same easy way of setup and maintenance of Windows operating system. The reason for this is because it can be done through an easy installation process of the system, removal of many complicated tasks that concerning on the

database administration, and also it use a graphical computing environment for every system administrator task.

- Microsoft SQL Server offers new or extended database capabilities, like sending and receiving message, and also managing login security.

Beside those benefits, Microsoft SQL Server also has several important aspects, including:

- SQL Server is easy to use.
- SQL Server can be scaled from a mobile laptop to symmetric multiprocessor (SMP) system.
- SQL Server provides the user with business intelligence features, which is become an important aspect, because until now it can only be accessed through Oracle or other more expensive DBMSs [16].

## 2.2 Theoretical Frameworks

The basic framework that will be used for this application is Microsoft Solution Framework for Agile Software Development (MSF Agile). MSF basically is a set of models and concepts that come from Microsoft, which enable the user to apply it into their IT projects, including the development of the application, networking, or infrastructure projects.

MSF Agile is suitable for Visual Studio Team System because it supports the development of the application by continuing the process of learning and

refinement. This thesis is going to use Visual Studio as the tool to make the HR application as well. That is why this thesis choose to use MSF Agile as the basic framework [1].

### **2.2.1 Microsoft Solution Framework Agile Concept**

In MSF for agile software development, each team member has their own role. They can have one or more roles, and work together as a team of peers. Inside those roles they will do several activities. Those activities can be grouped into work streams. Work product may be produced from those activities, which will be required for the performance of several states. Things that are included as the work products are the project plans, project documents, project source code, and the other output as the result of the activities.

Everything that is done inside the team project will be recorded into the database, which is called as the work items. The reason to do this recording is to keep track the state of activities that is done within the team project in real time [1].

### **2.2.2 Microsoft Solution Framework Agile Method**

There are several phases that included in this MSF method, which suitable for people that going to use Visual Studio as their tool to make the application. Those methods in MSF Agile are [1]:

### 1. Envisioning Phase

In this phase, the team should define the clear goals about the project, what are the things that we want to achieve at the end of the project, the vision and mission, and also the constraints of the project. For this thesis, it will only cover about the recruitment and selection process.

### 2. Planning Phase

In this phase, the team will define the specification, including the design process, planning for the project, cost project, and also schedule of the project. The team should do the solution planning to the application, which suits the goals and the constraints that will fulfill the vision and mission of the project.

### 3. Developing Phase

In this phase, the team creates the project, including make the documentation and test plans. The team will configure and create the realization of the application, based on the solution to develop that system.

### 4. Stabilizing Phase

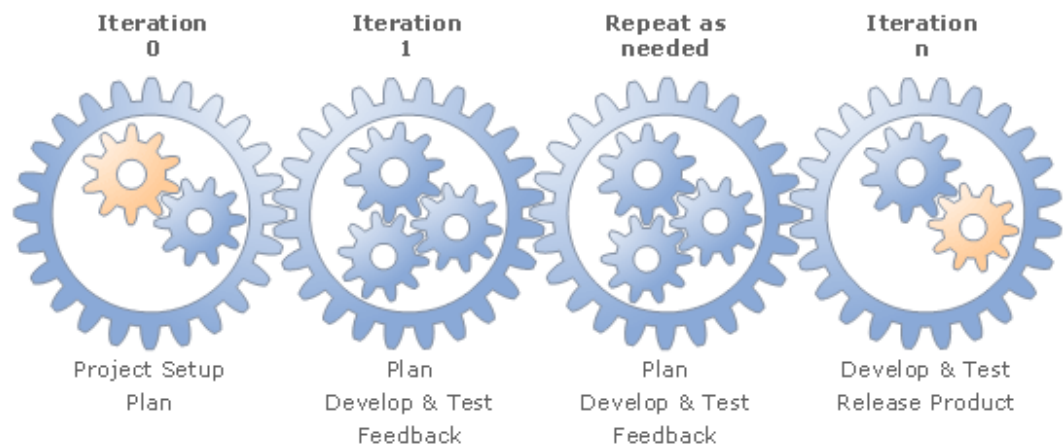
In this phase, the team will do the testing activity to the complete application. Usually, the team will emphasize on the usage and operation in the real condition. They can be tested in many kind of scenarios to find and solve the bugs, the lack factor, or other factors that influence the performance of that application.

## 5. Deploying Phase

In this phase, the team develop the operations and support by review the project or use the user satisfaction survey. After that, the user have to give the approval for the final product. It is also possible to see the further development of that project.

### 2.2.3 Microsoft Solution Framework Agile Cycles and Iterations

As stated before, this MSF Agile is suitable for developing a software or application using Visual Studio, because it supports the way of making the application by continuing the process of learning and refinement. MSF Agile uses several cycles and iterations in the process.



**Figure 6 Microsoft Solution Framework Agile Cycles and Iterations**

These cycles and iterations are including the process of product definition, development, and testing. The reason for MSF Agile to give the method for these cycles and iterations is that by having several cycles



and iterations, people can increase the completion of the application. Beside that, by having several small cycles and iterations, people can reduce the margin of error and get more accurate result for our project plans. However, project iterations usually focus on another part of the application as the project approaches release [1].

#### **2.2.4 Microsoft Solution Framework Agile Governance**

MSF Agile is also using Governance, which become the part that concern more on the control for time and money of the application. In this Governance step, there will be 5 questions as its checkpoint. Those questions are:

1. Objectives Reviewed

The question for objectives reviewed is: “Will we meet the minimum acceptance level of functionality?”

2. Progress Assessed

The question for progress assessed is: “Are we making the necessary progress to meet our deadline?”

3. Test Thresholds Evaluated

The question for test thresholds evaluated is: “Are we maintaining an appropriate level of quality?”

4. Risks Identified and Mitigated

The question for risks identified and mitigated is: “Are we addressing project and technical risk?”

## 5. Deployment Ready

The question for deployment ready is: “Have we designed and created a solution that is ready to deploy?” [1].

### 2.2.5 Microsoft Solution Framework Agile Principles

MSF Agile is a agile process to help people in developing the software, which build using .NET or other object – oriented applications. In order to reach the goal of the development software, there are several principles in MSF Agile, such as [1]:

#### 1. Partner with Customers

The MSF team make the decision based on customer and understand what customer want. MSF team wants to make the customer become active in the participation of project delivery. A great team will focus on customer satisfaction.

#### 2. Work Toward a Shared Vision

MSF strongly work based on the shared vision in order to reach the project goal. Shared vision ensures all of the team members agree to all of the accomplishment on building the project.

#### 3. Deliver Incremental Value

By having the frequent delivery, it is proved that the process and infrastructure of the project will be improved. By doing this, we can find the errors, bugs, risks, or any other requirements that we may miss before.

#### 4. Invest in Quality

To become a successful team, every member inside the team should feel responsible to the quality of the product. We should make a plan and schedule in order to build the quality of product. By doing this, it will reduce the velocity, which may provide enough slack in the future of iteration. This will be the important thing to help us in reduce the bugs.

#### 5. Empower Team Member

Each member of the team should be empowered to deliver on their commitments and have their own confidential. The customer has the rights to make sure that the team will meet its commitments and plan on the basis. It is also important to tell the customer about every delay and change in the project, as soon as possible.

#### 6. Establish Clear Accountability

In MSF agile team, every role has equal weighted goals, and there is no single individual is able to become successfully represent all of the different goals. The way to solve this problem is to have the team of peers, which will combine a clear line of accountability, together with the shared responsibility to the stakeholders.

#### 7. Learn from All Experiences

In MSF agile, each project and iteration will become the opportunity to learn about the new things, as long as they have the honest feedback and reflection. There will be also an opportunity

to implement the proven practices of others and schedule the time line for learning.

#### 8. Foster Open Communication

MSF agile provide an open and honest approach of communication, either within the team project or with the stakeholders. The free flow of information will give the opportunity to all of the team member in order to give their contribution on reducing uncertainties within the project.

#### 9. Stay Agile, Adapt to Change

MSF agile ensures that all of the core role in the team project will be available along the project to contribute their decision to the changes. If there is any new challenges arise, MSF team will fosters agility to address the issues.

### **2.2.6 Microsoft Solution Framework Agile Mindsets**

A mindset in MSF agile is collection of value, which determine how a person work as an individual and their respond to the situations. There are several mindset inside the MSF Agile, such as [1]:

#### 1. Focus on Business Value

Each software syste is built to solve a business problem. A software development project will has its value based on the addition of its business value. This business value should be keep to become the focus of the project.

## 2. Advocate for Your Constituency

There are many stakeholders in developing the project, which has their own interest in the project. We have to remember and understand whose interest we represent in our project.

## 3. Take Pride on Workmanship

Taking pride in the contribution of the solution is one of the important part in order to creating the quality of the product. The result of this pride is motivation and sense of responsibility. All of the roles in MSF agile has their own responsibility in building the product with the best quality.

## 4. Deliver on Your Commitments

In MSF agile, the team member should be the agile person. It means that the team member should be active in attacking the work on which other are dependent, and delivering it as soon as possible. This will help team member to empower their teammates.

## 5. Look at the Big Picture

MSF agile has a mindset to see the way of every piece in the solution will fit into a broader solution. Looking at the big picture also means to be more focused on execution and what will be the deliverable in the end of the project, and also be less focused on the process of getting into there.

## 6. Foster a Team of Peers

This mindset means that every advocacy and role has equal value. The keys to fostering a team of peers are unrestricted

communication between each role of team project and project transparency. The result of this mindset are increased in team accountability, have the effective communication, and teamwork.

7. Practice Good Citizenship

This mindset focuses on stewardship of corporate, project, and computing resources. Citizenship makes themselves to conduct the project in an efficient manner with many ways in order to optimize the system resources.

8. Learn Continuously

Willingness to learn allows the team members to benefit from the lessons learned by making mistakes. Teams that has willingness to learn, review, and retrospectives create the way of improvement and continuing success.

9. Internalize Qualities of Service

This mindset focuses at the solution and the developement of the plan based on the customer experience. By using this mindset, we look at the big picture and turn the implicit quality of service requirements.