Chapter 3 Research Methodology

A system development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system (CMS of United States 2008). It is a model to define the set of activities that leads to a system development (Sommerville 2004). Choosing an effective and ideal development methodology is a very important step in order to ensure the project development done on time and reduce the risk. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses (CMS of United States 2008). One system development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations (CMS of United States 2008). This chapter describes and explains the research methodology used in this dissertation. Please be informed that most of the contents and explanations of this chapter are referred to the study of (CMS of United States 2008), (Pressman 1997) and (Sommerville 2004).


### 3.1 Waterfall Model

The waterfall model is a systems development life cycle model for software engineering (Sommerville 2004). This model takes the fundamental process activities of specification, development, validation and evolution and represents them as separate process phases. It describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. This model can use to emphasis on planning, time schedules, target dates, budgets and implementation of an entire system at one time (CMS of United States 2008). The strengths and weaknesses of waterfall model are summarized in Table 3.1.

![Waterfall model diagram](Sommerville 2004)

**Figure 3.1: Waterfall model diagram (Sommerville 2004).**
Table 3.1: *Summary of strengths and weaknesses of waterfall model (CMS of United States 2008)*

<table>
<thead>
<tr>
<th>Strength</th>
<th>Weakness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal for supporting less experienced project teams and project managers.</td>
<td>Inflexible, slow, costly and cumbersome due to significant structure and tight controls.</td>
</tr>
<tr>
<td>Orderly sequence of development steps and strict controls.</td>
<td>Project progresses forward, with only slight movement backward.</td>
</tr>
<tr>
<td>Progress of system development is measurable.</td>
<td>Problems are often not discovered until system testing.</td>
</tr>
<tr>
<td>Conserves resources.</td>
<td>Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged.</td>
</tr>
</tbody>
</table>

### 3.2 Incremental Model

Incremental model is a model combine linear sequential model (waterfall model) with iterative philosophy (prototyping model). The primary objective of this model is to reduce inherent project risk by breaking a project into smaller segments and providing more ease of change during the development process (CMS of United States 2008; Sommerville 2004). Overall requirements are defined before proceeding to evolutionary, mini waterfall development of individual increments of the system. Under this model, a series of mini waterfalls are performed, where all phases (requirements, analysis, design, implementation
and testing) of the waterfall development model are completed for a small part of the system, before proceeding to the next increment (CMS of United States 2008; Pressman 1997). The strengths and weaknesses of incremental model are summarized in Table 3.2.

Figure 3.2: Incremental model diagram (Sommerville 2004).
Table 3.2: Summary of strengths and weaknesses of incremental model (CMS of United States 2008)

<table>
<thead>
<tr>
<th>Strength</th>
<th>Weakness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders can be given concrete evidence of project status throughout the life cycle.</td>
<td>Mini-Waterfalls utilizing usually lack of overall consideration of the business problem and technical requirements for the overall system.</td>
</tr>
<tr>
<td>Helps to mitigate integration and architectural risks earlier in the project.</td>
<td>Some modules will be completed much earlier than others.</td>
</tr>
<tr>
<td>Allows delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.</td>
<td>Difficult problems tend to be pushed to the future.</td>
</tr>
<tr>
<td>Ability to monitor the effect of incremental changes, isolate issues and make adjustments before the organization is negatively impacted.</td>
<td></td>
</tr>
</tbody>
</table>

3.3 Research Methodology of this dissertation

The development model chosen for this dissertation is based on incremental model. Incremental model has a series of mini waterfall performed, where all phases (requirements, analysis, design, implementation and testing) of the Waterfall development model are completed for a small part of the system, before proceeding to the next increment. The overall requirements are defined before proceeding to evolutionary in mini waterfall development of individual increments of the system. The initial software concept,
requirements analysis, and design of architecture and system core are defined using the waterfall approach, followed by iterative prototyping approach on the final prototype which involves system testing, system operation and maintenance.

### 3.3.1 Requirement definition

The requirement definition of this dissertation is to identify the research scope and limitation. In this dissertation, there are seven design principles (Refers to Section 4.1) and four design assumptions (Refers to Section 4.2) need to be considered.

### 3.3.2 System and software design

The proposed anomaly activity detection framework is design to function in the application layer of ZigBee. The proposed framework interacts with the application framework of sensor node to detect abnormal sensor network activity in ZigBee communication between sensor networks. There are 3 main processes in the proposed framework, which are features extraction (Refers to Section 4.4.1), Naïve Bayes classifier (Refers to Section 4.4.2) and modeling sensor network traffic activity behavior (Refers to Section 4.4.3).

Features extraction phase extracted the sensor node traffic activity for traffic classification. Follow with Naïve Bayes classifier involved in converting sensor network traffic activity frequency into the calculation metric. Lastly, the proposed framework modeled sensor
network traffic activity behavior to construct a threshold through a learning phase (Refers to Section 4.4.3.1). Then the detection phase will perform anomaly activity detection through comparing current traffic metric with the threshold metrics (Refers to Section 4.4.3.2).

### 3.3.3 Implementation

The proposed anomaly activity detection framework developed in Dynamic C and embedded into RabbitCore RCM4500W microprocessor in a ZigBee communication environment. The deployment of the proposed framework is in an ad hoc manner. The proposed detection framework is a local detection framework and did not rely on the knowledge of sample attacks because the goal for this dissertation is to detect anomaly activity (suspicious activity) in sensor network.

### 3.3.4 System testing

To identify the effectiveness and efficiency of the proposed anomaly activity detection framework, a series of experiments have been undergoing. In the experiment, a wireless sensor network with six sensor nodes has been established in an ad hoc architecture. Five established sensor nodes in the experiments had been chosen as the launching platform to perform Denial of Service (DoS) attacks. One sensor node chosen as the ‘victim’ to the
Denial of Service (DoS) attacks. The goal of this experiment is to identify the detection rate and number of false positives of the proposed framework.

3.4 Research Activities

This section explains how the research activities carried along the development of anomaly activity detection framework with artificial immune system approach in sensor network.

3.4.1 Software Specification

Software specification is a process of understanding and identifying the requirements and constraints in system development. This is a particular critical stage as errors at this stage may lead to later problem in system design and implementation (Sommerville 2004).

In this research, the software specification focuses in the review of literature (Chapter 2). First, the investigation had carried on the application domain of this research which is wireless sensor network (WSN). Wireless sensor network shares some similarity with Ad Hoc wireless network (Refers to Section 2.1.1). But there are some difference between wireless sensor networks and Ad Hoc networks (Refers to Section 2.1.2). The architecture of wireless sensor network (Refers to Section 2.1.3) also plays an important role in the applications of wireless sensor network (Refers to Section 2.1.4).
Chapter 3 Research Methodology

The human immune system (Refers to Section 2.2) approach which inspired the artificial immune system (Refers to Section 2.3) also had some reviews and studies. The review of literature carried on by investigating the traffic activity in wireless sensor network (Refers to Section 2.4). The reviews had categories into general traffic classification (Refers to Section 2.4.1), naïve bayes and bayesian classifier (Refers to Section 2.4.2), artificial immune systems inspired classification (Refers to Section 2.4.3), anomaly activity in wireless sensor network (Refers to Section 2.4.4) and denial of service attacks in wireless sensor network (Refers to Section 2.4.5).

In order to obtain the ideas of anomaly activity detection framework in sensor network, the literature review had carried on by focus in intrusion detection review (Refers to Section 2.5), two intrusion detection which is misuse detection (Refers to Section 2.5.1) and anomaly detection (Refers to Section 2.5.2), lastly follows by related works in anomaly activity detection of wireless sensor network (Refers to Section 2.5.3).

3.4.2 Software Design and Implementation

The software design and implementation stage is the process of converting a system specification into an executable system. This stage usually involved the programming. The software design is a description of the structure of the software to be implemented, the data part of the system and the algorithms used. The design process may involve developing several models of the system at different levels of abstraction (Sommerville 2004).
In this research, the software design process is based in a Dynamic C programming environment. The proposed anomaly activity detection framework with artificial immune system approach had developed by using Dynamic C programming language. The Dynamic C programming language is a programming language used to code RabbitCore microprocessor. The proposed anomaly activity detection framework with artificial immune system approach had embedded into RabbitCore RCM4500W microprocessor.

The RabbitCore microprocessor is a C-Programmable module with supporting Supports ZigBee/802.15.4 protocol in a mesh topologies. ZigBee is a communication protocol which targets the application domain of low power, low duty cycle and low data rate requirement of devices. To evaluate effectiveness and efficiency of this proposed system, the RabbitCore RCM4500W microprocessor had been used to demonstrate a wireless sensor network communication environment.

### 3.4.3 Software Validation

Software validation generally is a process to show that a system conforms to its specification and meets the expectations of the requirement definition and program development (Sommerville 2004). In this research, six sets of testing will be used to evaluate effectiveness and efficiency of the proposed system. The Denial of Service (DoS) attacks will be performed as the resource
exhaustion attacks in wireless sensor network. In this testing process, sensor nodes (protected resources) act as self pattern while Denial of Service (DoS) attacks act as non self pattern. The detection module in each sensor node will go through a learning process to determine the threshold for a self behaviour (normal activity). After determine the threshold for a self behaviour, the sensor can use to detect anomalous (anomaly activity) behaviour.

3.5 Hardware and Software Consideration

This section summarizes the hardware, software and tools that had been chosen to be used in this project.

3.5.1 RabbitCore RCM4500W microprocessor

The RCM4510W is a type of RabbitCore modules which add ZigBee/ 802.15.4 functionality to the existing Rabbit 4000 microprocessor. RCM4510W supports ZigBee/802.15.4 point-to-point, point-to-multipoint, and mesh topologies (RabbitCore RCM4500W User’s Manual 2007). These features allow developer to create a low-cost, low-power, embedded wireless control and communications system. The RCM4510W is programmed using version 10.11 or later of Dynamic C. RabbitCore modules are products from rabbit semiconductor owned by Z-World, who sells compilers, assemblers, and development tools for embedded systems (Pearson 2007).
An RCM4510W with no loading at the outputs operating at 29.48 MHz typically draws 80 mA, and may draw up to 150 mA while the ZigBee modem is transmitting or receiving. The RCM4510W requires a regulated 3.3 V DC ±5% power source (RabbitCore RCM4500W User’s Manual 2007).

The ZigBee modem included with the RCM4510W presently supports using the RCM4510W RabbitCore module in a mesh network. RCM4510W modules are loaded with firmware at the factory to serve as routers or end devices; coordinator firmware is included in the Dynamic C installation (RabbitCore RCM4500W User’s Manual 2007). RCM4510W can function as a network coordinator, gateway, or end device.

RCM4510W Features (RabbitCore RCM4500W User’s Manual 2007):

- Small size: 1.84” × 2.85” × 0.54” (47 mm × 72 mm × 14 mm)
- Microprocessor: Rabbit 4000 running at 29.49 MHz
- Up to 40 general-purpose I/O lines configurable with up to four alternate functions
- Up to 9 additional general-purpose I/O lines (up to four of which may be set up as analog inputs) available through the ZigBee modem
- 3.3 V I/O lines with low-power modes down to 2 kHz
- Six CMOS-compatible serial ports — four ports are configurable as a clocked serial port (SPI), and two ports are configurable as SDLC/HDLC serial ports.
- Alternate I/O bus can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), I/O read/write
- 512K flash memory, 512K data SRAM
- Real-time clock
- Watchdog supervisor

Figure 3.3: *RCM4500W architecture* (RabbitCore RCM4500W User’s Manual 2007).

### 3.5.2 Digi XBee USB

Digi XBee USB is a device used as ZigBee coordinator for RCM4510W RabbitCore module in a mesh network. RCM4510W modules are loaded with firmware at the factory to
serve as routers or end devices in a mesh network. The XBee USB Coordinator is set up using XBee X-CTU (Refers to Section 3.5.4) to allow devices to join the personal area network (PAN) in a ZigBee communication environment. The XBee End Device is configured via X-CTU to automatically join the PAN.

### 3.5.3 Dynamic C

Dynamic C is an integrated development system for writing embedded software. It is designed for use with Rabbit controllers and other controllers based on the Rabbit microprocessor (Dynamic C User’s Manual 2006). It is designed for use with Z-World controllers and other controllers based on the Rabbit Microprocessor (Introduction to Z-World’s Dynamic C for Rabbit Microprocessors 2004). Dynamic C also supports assembly language programming. Dynamic C programs can be executed and debugged interactively at the source code or machine code level. Dynamic C compiles directly to memory. On a fast PC, Dynamic C might load 30,000 bytes of code in 5 seconds at a baud rate of 115,200 bps (Dynamic C User’s Manual 2006).

Dynamic C comes with many function libraries, all in source code. These libraries support real-time programming, machine level I/O, and provide standard string and math functions. They are located in the LIB subdirectory where Dynamic C was installed. The default library file extension is .LIB. Dynamic C uses functions and data from library files and
compiles them with an application program that is then downloaded to a controller or saved to a .bin file.

Although Dynamic C uses .lib as the library extension, the programmer may use anything they like as long as the complete path is entered in your LIB.DIR file. There is no way to define file scope variables in Dynamic C libraries. LIB.DIR contains pathnames for all libraries that are to be known to Dynamic C. The programmer can add or remove libraries from this list. The factory default is for this file to contain all the libraries on the Dynamic C distribution disk. Any library that is to be used in a Dynamic C program must be listed in the file LIB.DIR, or another *.DIR file specified by the user (Introduction to Z-World’s Dynamic C for Rabbit Microprocessors 2004).

In Dynamic C, a project is an environment that consists of opened source files, a BIOS file, available libraries, and the conditions under which the source files will be compiled. A project maintains a compilation environment in a file with the extension .dcp. An application program (the default file extension is .c) consists of a source code file that contains a main function (called main) and others are user defined functions (Introduction to Z-World’s Dynamic C for Rabbit Microprocessors 2004). The result of the application execution will show in a Dynamic C’s STDIO window (Best 2007).
3.5.4 XBee X-CTU

XBee X-CTU is a personal computer application used to configure and test the XBee modules via a personal computer’s COM port. X-CTU also incorporates with a built in terminal emulator function (Best 2007). XBee X-CTU can use to configure the XBee I/O module, or update RabbitCore module’s firmware by downloaded directly to the RabbitCore module via MaxStream’s X-CTU utility (RabbitCore RCM4500W User’s Manual 2007).

Rabbit firmware is written using Rabbit Semiconductor’s Dynamic C 9.25 compiler, which also integrates the Rabbit microprocessor’s debugging environment. The XBee application firmware that runs on the Rabbit microprocessor is based on the XBee AT command set (RabbitCore RCM4500W User’s Manual 2007).

There are 2 types of firmware for RCM4500W RabbitCore module:

- Firmware of the type XB24-B_ZigBee_11….ebl is used for a coordinator RCM4510W.
- Firmware of the type XB24-B_ZigBee_13….ebl is used for an end device/router RCM4510W.
3.6 Hardware and Software Requirement

There are hardware requirement and software requirement.

3.6.1 Hardware Requirement

The hardware requirement for this project is:

1) Personal Computer with below specifications:
   - Intel Pentium 4 Processor (1.6 Hz and above)
   - At least 256 MB of RAM (512 MB is recommended)
   - Hard disk space of 10 GB

2) RabbitCore RCM4500W microprocessor

3) Digi XBee USB

3.6.2 Software Requirement

The software requirement to run this system is:

1) Microsoft Windows XP Professional (Service Pack 2 and above)

2) Dynamic C

3) XBee X-CTU
3.7 Summary

The development model chosen for this dissertation is based on incremental model to ensure the research project is on track and take corrective action as necessary in order to achieve the objectives in the proposed scheme. Incremental model has a series of mini waterfall performed, where all phases (requirements, analysis, design, implementation and testing) of the Waterfall development model are completed for a small part of the system, before proceeding to the next increment. The first phase is the requirement definition where the research scope and limitation is identified. In this dissertation, there are seven design principles and four design assumptions need to be considered. The second phase is system and software design where the 3 main processes in the proposed framework are discussed. There are features extraction, Naïve Bayes classifier and modeling sensor network traffic activity behavior. In the third phase, the implementation of the proposed anomaly activity detection framework developed in Dynamic C and embedded into RabbitCore RCM4500W microprocessor. Lastly, the fourth phase is the system testing to identify the effectiveness and efficiency of the proposed anomaly activity detection framework, a series of experiments have been undergoing.