4-2013

An Automated Approach to Dynamic Site Layout Planning

Duy Huu Nguyen
Western Michigan University, duy.huu.ng@gmail.com

AN AUTOMATED APPROACH TO DYNAMIC SITE LAYOUT PLANNING

by

Duy Huu Nguyen

A Thesis submitted to the Graduate College
in partial fulfillment of the requirements
for the Degree of Master in Engineering (Civil)
Department of Civil and Construction Engineering
Western Michigan University
April 2013

Thesis Committee:

Abiola Akanmu, Ph.D., Chair
Haluk Aktan, Ph.D.
Yufeng Hu, Ph.D.

AN AUTOMATED APPROACH TO DYNAMIC SITE LAYOUT PLANNING

Duy Huu Nguyen, M.S.E.

Western Michigan University, 2013

Dynamic site layout planning involves determining the optimum location of resources on the construction site. Site layouts are considered dynamic because construction activities and the number and nature of associated resources change as the project progresses. Previous research has focused on the use of several algorithms to address dynamic site layout problems. These efforts do not enable automated tracking of the dynamic context of construction resources, site space and activities. This research presents an automated adaptive component level system capable of generating layouts which are optimized throughout the phases of construction projects. The system involves the integration of building information models, genetic algorithm and the radio frequency identification real-time location sensing system. The developed system is based on sensing components and construction activity status, path predictability using genetic algorithm, cost prediction and site layout model generation. A practical implementation is presented to demonstrate the functionality and utility of the developed system. The benefits of this system include access to real-time status of resources and site spaces which will aid the construction project team in quick decision making. The system also has potential for enhancing real-time communication and collaboration between the personnel on the job site and the office.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Abiola Akanmu, for her endless effort and instructions. She is such a great advisor as well as mentor. It would have been impossible for me to complete my graduate studies without her exceptional guidance. Her dedication and encouragement is sincerely appreciated.

Secondly, I would like to thank my committee professors: Professor Haluk Aktan and Professor Yufeng Hu. Their advices and comments on my graduate research were very informative, helpful and highly appreciated. I also want to thank Professor Laila Cure for giving me instructive direction and endless encouragement toward my goal.

Lastly, I would like to thank my family and friends: Cheekian Teng, Syed Hammad Rasheed and Anh Nguyen who were always there to support and encourage me in every possible ways. Without them, there would have been a missing piece of this research.

Duy Huu Nguyen

TABLE OF CONTENTS

Table of Contents - Continued

CHAPTER

Table of Contents - Continued

CHAPTER

Table of Contents - Continued

LIST OF TABLES

List of Tables - Continued

LIST OF FIGURES

List of Figures - Continued

CHAPTER 1: INTRODUCTION

1.1. Background

There have been significant attempts to improve various aspects of the construction project delivery process through the integration of innovative information and communication technologies (such as building information modeling, wireless sensing and other data acquisition technologies) with the construction process. These have resulted in incremental changes with marginal level of improvement; thus, there are still opportunities for improvement. A potential area still requiring attention is site layout planning. Site layout planning involves effectively identifying and positioning temporary facilities during construction. Site layout planning has tremendous implications on the cost, productivity and safety on construction sites. This chapter introduces the research project by describing the problem or issues with existing site layout planning process. This chapter also presents the aim and objectives of the research, the research process, which was followed, and an overview of the structure of the thesis.

1.2. Problem Statement

Space is a major resource required on construction sites for accommodating various types of materials and equipment. Efficiently allocating this resource throughout the duration of a project is a critical concern for construction planners as it has a significant impact on cost, productivity and safety of construction operations (Ning et al. 2010). Planning the layout of site spaces involves positioning temporary facilities needed

to support various construction activities and tasks during construction. Such temporary facilities include site offices, storage areas, staging areas, workshops, and parking areas. Due to the complexity of this process, construction managers usually perform site layout planning based on previous experience or first-come-first-serve basis, thus leading to ambiguity and inefficiency (Mawdesley et al. 2002). During construction, depending on the type and duration of task performed, the requirements of the site layout changes, as such, site layout has been modeled by a number of researchers as a dynamic problem (Lam et al. 2007; Lien and Cheng 2012; Ning et al. 2010). The dynamic behavior of a site layout requires knowledge of space availability and consideration of possible reuse of space and relocation of temporary facilities (trailers, office, etc.) while also minimizing the associated layout cost throughout the duration of a project. Facilities constrain one another when determining their final position. For example, inflammable substances must be kept at a minimum distance from the building, the equipment location cannot overlap the storage area, the crane needs to be positioned in close proximity to the staging area and the excavator and excavation area needs to be kept close. These requirements change as the activities and project phases change.

Furthermore, to generate site layouts, project managers need information about the availability of the site spaces i.e. to know which spaces are available to place temporary facilities. Presently, project managers spend significant amount of time walking around the job site to visually assess free spaces. For example, if the site layout is scheduled to be generated after the first phase, the project manager will need to walk around the site to determine if the initially assigned spaces are vacant. The availability of the site spaces depends on the status of the associated task or activity. A site space may

be vacant either because an ongoing activity requires more materials (which are scheduled to arrive) or the associated activity has actually ended. This implies that effective site layout planning depends on the project schedule, project status and real-time knowledge of site availability. Existing efforts have only concentrated on generating site layouts based on the project schedule and phase. As such, the true dynamic nature of site layouts has not been fully addressed.

1.3. Research Objectives

In order to address the problems identified above, the research aims to develop an adaptive automated component level system for construction site layout generation by using Genetic Algorithm as optimization method. The specific objectives are:

- To identify the requirements for effective site layout generation. This requirements will be developed through an extensive review of literature and informal interactions with project managers;
- To investigate the potential of genetic algorithm (GA) for generating the optimum layout based on the identified requirements. This will include formulating the site layout problem. The formulated problem will be implemented in GA and validated using a hybrid swarm approach;
- To develop a systems and software architecture for the adaptive automated component level site layout generation system;
- To develop and experiment with automated site layout generation on laboratory scale prototypes. This experiment will enable tracking the status of

key components and automatic generation of site layouts using the laboratory scale physical prototype;

- To test and validate the developed prototype on a sample case study.

1.3.1. Research Scope

The specific focus of this study is on the use of sensing technologies and computational algorithm such as Genetic Algorithm for automated generation of construction site layouts. This was initially explored by investigating the requirements for site layout generation through detailed analysis of literature and interactions with industry experts. The requirements provided a basis for formulating the site layout problem, which was implemented using a GA approach. The GA approach was validated with a previous hybrid swarm formulation. A prototype system was developed and implemented based on integrating sensing technologies and the formulated GA.

1.3.2. Research Contribution

The results of this study have provided a foundation for improving the construction site layout planning by demonstrating the technical capability of sensing technologies and computational algorithms for generating optimum site layouts. The outcome of this research provides opportunities for future comprehensive system using other image-based technologies. The detailed contributions of the research are contained in Chapter 4 and 5 – a brief summary is presented here:

- Key requirements for construction site layout planning;

- Technologies and techniques for enhancing automated real-time site layout generation. They provide a basis for future research into how these technologies and techniques can be collectively integrated to develop a more comprehensive system approach for site layout generation;
- An automated real-time site layout information system, which integrates building information model, Genetic Algorithm, RFID-RTLS system and physical construction components, has been developed. This system has the following capabilities: real-time component and schedule tracking and site layout generation. The adoption of this system has the following implications for the construction industry:
    - Reduction of uncertainty and unpredictability because the activities and processes can be more closely monitored and controlled;
    - Reduction of construction project delivery times and costs. An example of this will be a reduction in site layout generation cost due to improved tracking and visualization of cost effective site layout options.

1.4. Thesis Structure

This thesis is organized as follows (as shown in Figure 1-1): the next chapter is literature review and followed by the detail introduction of computational approaches to site layout planning, tracking technologies and BIM. Chapter 3 presents the approach adopted in this thesis. Chapter 4 presents the system development while chapter 5 discusses the system validation and case studies. The final chapter concludes this thesis

by summarizing the work and providing the final conclusions. A more detailed description of the contents of each chapter is as follows:

Chapter 1: Introduction

This chapter presented the general introduction and discusses the background to the research. It discusses the need for the research and outlines the aim and objectives. It also provides a structure of the thesis.

Chapter 2: Literature Review

This chapter begins by discussing site layout planning problems from the static and dynamic points of view. Based on this, the chapter highlights the requirements for site layout planning. In addition, this chapter presents a comparison between different optimization algorithms used for site layout planning. Sensing technologies and their suitability for use for construction tracking applications is discussed. The chapter concludes by presenting limitations and opportunities for improvement of current site layout planning process.

Chapter 3: Research Methods and Approach

The problem formulation and optimization algorithm utilized in this study are presented in more detail in this section. This chapter also presents the role of BIM and the sensing technology used for the study.

Chapter 4: Prototype System Development

This chapter describes the development of the prototype used for the study. This includes an illustration of the flowchart, software and system architecture.

Chapter 5: Prototype System Validation and Case study

The validated of the optimization algorithm is presented in the chapter. Sample application and implementation of the developed system on construction projects are described in this chapter.

Chapter 6: Summary and Conclusions

This chapter begins by summarizing the work performed. Final conclusions are then presented along with the recommendations for future.

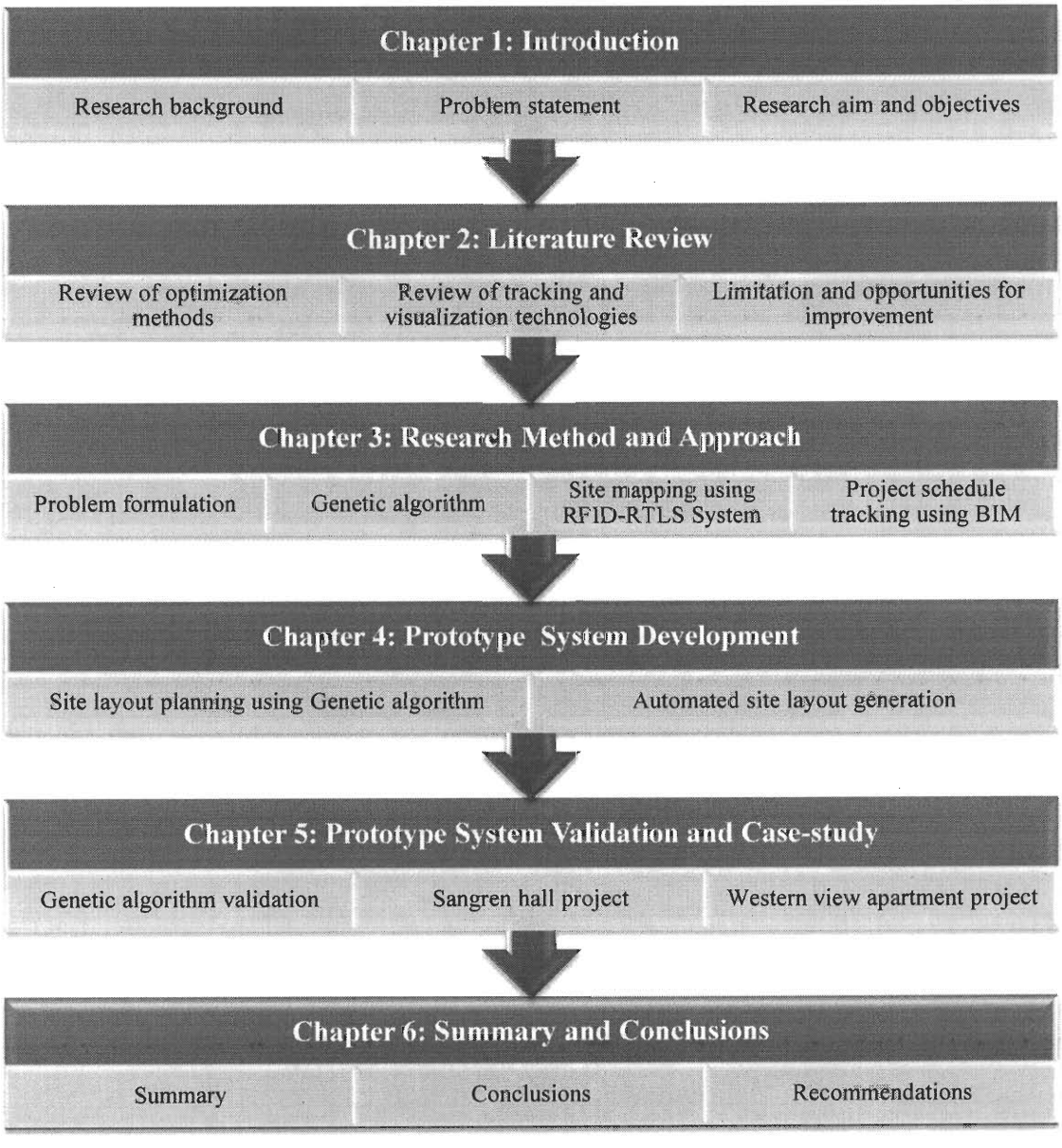


Figure 1-1 – Thesis structure

CHAPTER 2: LITERATURE REVIEW

2.1. Introduction

There have been a number of research efforts to improving site layout planning. These efforts spurred from the use of optimization techniques for variety of applications in the construction industry. There have also been waves in the use of visualization and sensing technologies for variety of tracking applications. This chapter presents various optimization based efforts to site layout planning. This chapter discusses the different enabling technologies utilized in the construction industry such as wireless sensors and other data acquisition technologies, photographs, laser scanners, photogrammetry, communication networks and mobile devices. This chapter concludes by highlighting opportunities for improvement on existing work.

2.2. Site Layout Planning

Site Layout Planning is categorized as a static and dynamic problem. These are briefly described as follows:

2.2.1. Static Site Layout Problems

Static site layout problem is related to temporary facilities (such as equipment, site offices, trailer, etc.), and obstacles (such as trees, electricity pole, etc.). According to Sadeghpour et al. (2002), static models cannot be changed once the layout of the

construction site has been designed. However, natural behavior of construction site is not static but dynamic and design models should be able to address to such behavior.

2.2.2. Dynamic Site Layout Problems

During construction, depending on the type and duration of task performed, the requirements of the site layout changes, as such; site layout has been modeled by a number of researchers as a dynamic problem (Lam et al. 2007; Lien and Cheng 2012; Ning et al. 2010). The dynamic behavior of a site layout requires consideration of possible reuse of space and relocation of temporary facilities (trailers, office, etc.) while also minimizing the associated layout cost throughout the duration of a project. A number of researchers have attempted solving this dynamic problem using evolutionary algorithms such as Tabu Search, Bee Algorithm, Max-min Ant System and Genetic algorithm. For example, Abdinnour-Helm and Hadley (2000) used Tabu search which is a local search method to find optimal layout for multi-floor facilities. Ning et al. (2010) applied Max-Min Ant System, an algorithm based on ant colony optimization, to solve a dynamic multi-objective site layout planning problem. Lien and Cheng (2012) combined the ability of global and local search from Bee Algorithm and Particle Swarm Optimization, respectively into the development of proposed hybrid swam intelligence based particle-bee algorithm to solve this particular optimization problem. In recent years, Genetic Algorithm (GA) has been of increasing interest to research scientists. Amongst the existing evolutionary algorithm used in solving the site layout problems, GA is the most widely used (El-Rayes and Said 2009). GA is a search algorithm based on the mechanics of natural selection and natural genetics. The advantage of GA over other

algorithms is in its strong evolutionary process of finding the optimal solution by the operation of crossover, selection and mutation of parents' generation (Ning et al. 2010). GA efficiently exploits historical information to speculate on new search points with expected improved performance (Goldberg 1989). Li and Love (2000) used genetic algorithm to address site-level unequal-area facility layout problem. The objective was to minimize the total traveling distance of site personnel between facilities but the effect of probability of crossover and mutation on convergence was not investigated. Hegazy and Elbeltagi (1999) also applied GA to search for a layout that minimizes the total travel distance. Cheung et al. (2002) applied Genetic Algorithm to site pre-cast layout planning problem. In this particular case, the layout was generated based on the cost per unit distance for each resource on site. Instead of considering each material cost, this paper concentrates on determining the total layout cost based on project phases. Mawdesley et al. (2002) proposed the use of genetic algorithm to determine the positions of facilities on site to minimize a fitness function, which consisted of four cost components: material transportation cost, setup cost, removal cost, and personnel movement cost. In their study, the optimization procedure was sensitive to the relative costs assigned to setup and shifting facilities. The study set different cost at different regions on site, which are treated as restricted area for algorithm to place facilities there. Furthermore, determination of layout and project scheduling procedure would need to be carried out to prove optimality; also the demonstration of shifting facilities was missing. Chandratre and Nandurkar (2011) also tried to solve dynamic layout problem with Genetic Algorithm, but the cost of shifting the facilities was not validated. In this research, the

cost of moving temporary facilities between phases will be determined and validated with an actual construction project.

2.3. Optimization Techniques

In mathematics, computer science or engineering, mathematical optimization known as optimization is the selection of a best solution from a given set of available solution population. In the simplest case, an optimization problem has the objective of maximizing or minimizing a real function by inputting values from an allowed set. Depends on the types of the problem, there are many optimization methods that can be chosen.

2.3.1. Types of Optimization Techniques in Site Layout Planning

For many years, many researchers have been searching for appropriate optimization methods to address site layout planning. The following optimization methods have been considered: Hybrid swarm intelligence based particle–bee algorithm, Bee algorithm, Particle swarm optimization, Max-min ant system, Appropriate dynamic programming, Minimum Total Potential Energy Principle from Physics, Simulated annealing, Evolutionary programming and genetic algorithm, etc.

2.3.1.1. Bee Algorithm

The Bee Algorithm (BA) is population-based search algorithm, which imitate the food foraging behavior of swarm of honeybee. It uses neighbor search combined with random search for optimizing problems. During harvesting season, a number of scout bees are sent out to search for potential flower with rich nectar and pollen. More bees usually should visit flowers with more nectar and pollen. After searching, scout bees return to the hive and perform waggle dance (as shown in Figure 2-1) that contains 3 pieces of information: direction to the particular flower, distance from hive to the flower and the fitness rating. Based on this information, the hive sends out number of bees to that particular flower for efficient food harvesting. The group is led by those scout bees. On the other hand, it monitors the food level on that flower so it can adjust the information given on the next waggle dance. Since the scout bees monitor and adjust information based on available food level, the group of working bees can be changed according to that information. This means that in site layout planning, frequencies between facilities can be optimized based on task requirements. Also, since the scout bees can pick the flower, it can minimize the travel distance between facilities since they know the exact location. Pham et al. (2006) presented the original BA and applied to complex optimization problem such as site layout planning. The objective was to minimize the travel distance between site locations and frequencies of trips between facilities. The result showed BA outperformed other methods, but the drawback is the number of tunable parameters used and it required multiple trial runs.

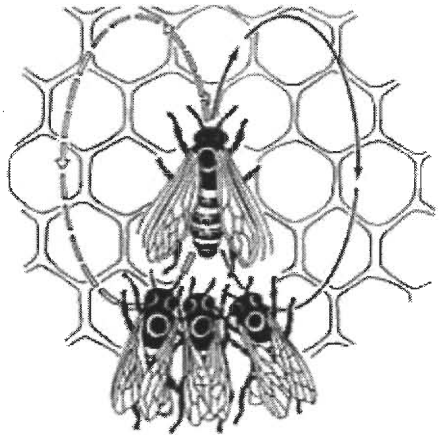


Figure 2-1 – Waggle dance of honey bees (Baykasoglu et al. 2007)

2.3.1.2. Particle Swarm Optimization

Particle Swarm Optimization (Maier et al. 2003) is a population-based optimization process which imitates the behavior of bird flocking, fish schooling or insect swarm (Figure 2-2). The process is also related to evolutionary programming and genetic algorithm, but it doesn't involve mutation or crossover properties. The method works by setting initialization swarm with random particle positions and velocities. For each particle, it evaluates the fitness and makes a span of schedule. If the current evaluated value is better than the previous, it sets to new value. The process finally changes the velocity and position of the particle. PSO has been quite popular for many researchers recently in solving site layout planning problem. The process is potentially used in local searching, but it converges early in highly discrete problems and traps into the local optimum solution.

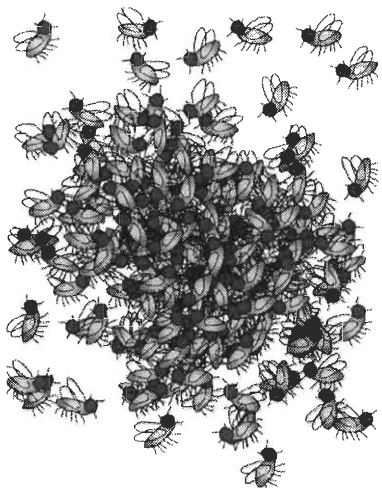


Figure 2-2 – Illustration of insect swarm (Eberhart and Shi 2004)

2.3.1.3. Hybrid Swarm Intelligence based Particle-Bee algorithm

Lien and Cheng (2012) combined BA and PSO into a proposed method called Hybrid Swarm Intelligence based Particle-Bee Algorithm (PBA). The algorithm is inspired by the intelligence behavior of bird and honeybee swarm and uses their advantages to solve site layout planning problem. It took advantage of global search of BA and local search of PSO. This algorithm differs from the others since it combines advantages of both algorithm and produces lowest mean and standard deviation of fitness based on travel distance and trip frequency. The objective is to minimize the travel distance between site locations and frequencies of trips between facilities but it neglects the practical factors such as work/information flow, safety and environment. Although PBA was proved to outperform BA and PSO, it is difficult to be modified to fit with other optimization problems.

2.3.1.4. Max-Min Ant System

Max-min Ant System from Ning et al. (2010) is one of Ant Colony Algorithm (ACO), which uses weight sum method to solve multi-objective optimization problem. The objective of implementation of this method is to minimize the total handling cost of interaction flow between facilities and the representative score of safety/environmental concerns. Safety/environmental concerns represent level of safety and environmental hazards, which may be caused by the closeness between 2 facilities. Also it could endanger workers by increasing the likelihood of accidents, noise, and pollution. While considering the closeness of facilities, there are factors that are taken into account: material flow, information flow, personnel flow, equipment flow, Safety/environmental concern, and user's preference. Based on these factors, weight factors are assigned to facilities in order to determine interaction relationship. The limitation of this method is that the overall solution may cover the solution quality for each objective function.

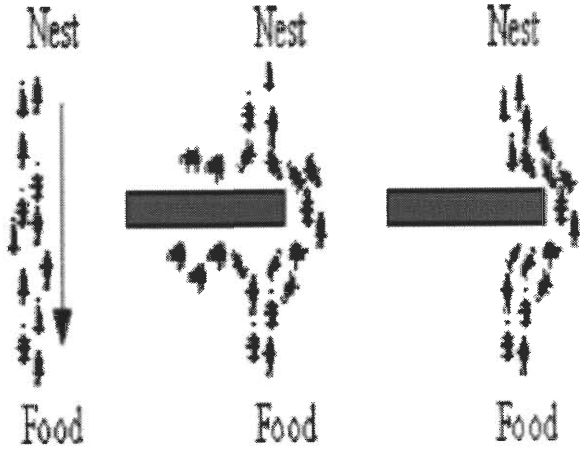


Figure 2-3 – Illustration of ant behavior (Selvi and Umarani 2010)

2.3.1.5. Approximate Dynamic Programming

El-Rayes and Said (2009) proposed dynamic site layout planning by using Approximate Dynamic Programming which is a technique for solving problem by breaking them into simpler smaller problems. This particular research uses approximate dynamic programming and chain of decision epochs to represent the positioning decisions of all facilities. The main objective of the paper is to minimize the total site layout cost by obtaining global optimum site layout plan over the entire project duration and to fulfill operational, organizational and safety requirements by modeling different geometric constraints between temporary facilities. The paper proved that the method can generate global optimal solution, but the limitations are: approximating resource travel paths as the shorted direct distance between site facilities, and assuming that site storage requirements are predetermined and static.

2.3.1.6. Minimum Total Potential Energy Principle from Physics

Andayesh and Sadeghpour (2011) implemented physics based principle into solving site layout problem. They proposed the method called minimum total potential energy principle (MTPE) from physics, which uses the energy principle governing a physical system in order to determine the optimum location of objects. The energy depends on internal forces and distance between particles. According to the law of conversation of energy, energy cannot be created or destroyed; it can only be transferred from 1 form to another. Total energy is equal to the sum potential energy and kinetic energy. The internal force acting on an object causes the movement and gain kinetic energy; which means that, it loses potential energy. The total potential energy decreases

gradually until the system reaches equilibrium state where all the forces are balanced and particles no longer move. In this method, internal force is referred as closeness weight and potential energy as objective function. The closeness weight (internal force) cause objects/temporary facilities (particles) to move, the potential energy (objective function) will decreases until the layout (system) is at equilibrium. For example, the ball and the earth have the potential energy, which is the function of the ball's weight and the distance to the earth. This method was then applied to solve site layout planning problem by minimizing travel distance and frequency of trips made by personnel.

2.3.1.7. Simulated Annealing

In mechanical principle, simulated annealing (Monte Carlo annealing) imitate the cooling process of metal annealing after have been heated up to the melting point. At high temperature, the atoms with great amount of energy moved arbitrary. As the temperature cools down, the atoms loose energy and gets close to the minimum point of the energy in the whole state space. The whole state space would be at its lowest point as the cooling process finishes. El-Gafy et al. (2010) implemented this principle to solve site layout planning problem with the objective of minimizing the frequencies of trips (in 1 day) made by personnel and travel distance between facilities. The problem starts out with the random creation of initial construction layouts, which is used as the starting point and current solution of the process. Then, the choice of cooling schedule is set. The formula then simulates the cooling schedule based on acceptable given probability. Once all parameters are set, the current layouts will be given a small perturbation to create new potential layouts. The process is repeated until the whole cooling cycle is iterated. As

temperature decreases, the algorithm becomes more inclined to only accept fitness-increasing changes. There is a significant tradeoff between the quality of the solutions and the time required to compute them in this method.

2.3.1.8. Evolutionary Algorithm

In artificial intelligence, Evolutionary Algorithm (EA) is a genetic population-based optimization algorithm, which inspired by the biology evolution (reproduction, mutation, recombination and selection). Fitness function then decides the environment within which the solution would live. A subset of EA is well-known Genetic Algorithm (GA), which mimics the similar concepts by using evolution operator such as mutation, crossover, selection or inheritance. These operators are explained in detail in the next chapter.

2.3.1.9. Genetic Algorithm

Genetic Algorithm (GA) has been known for a long time as population-based optimization process inspired by the biology evolution. GA works with an initial family of solutions known as current population. The current populations use genetic operators (such as mutation, crossover and selection) to develop potentially better solutions. Goldberg (1989) applied GA to solve site layout problem. Li and Love (1998) investigated different genetic operators and parameters to improve the performance of GA. Many modified GAs has been tested recently with the objective of minimizing the frequencies of trips (in 1 day) made by personnel and travel distance between facilities.

The main advantage of using GAs is that it only needs an objective function with no determination of problem space. However, the challenge is to find the appropriate problem representation and correct parameter of genetic operators.

2.3.2. Requirements for Effective Site Layout Planning

Based on the review of existing work, a number of requirements have been identified. These efforts are represented in Table 2-1.

From Table 2-1 and based on interaction with industry experts, the following have been identified as the factors influencing the site layout cost: expected frequency of trips between facilities, distances between available locations on job site, expected material transportation cost between facilities, shifting cost of facilities between locations and setup cost of facilities at any location. These are briefly described below:

Table 2-1 – Requirement for site layout planning

	GA	ADP	ACO	PBO	BA	PSO	SA	MTPE	EA
Minimizing the travel distance between locations/ facilities	X	X	X	X	X	X	X		
Minimizing material flow (unit per time unit)			X	X	X	X	X	X	X
Minimizing the cost for resource flow (material transportation cost)	X	X							
Minimizing information flow (communication between facilities)	X	X							
Minimizing the equipment flow (transfers material from 1 to another)			X	X	X	X			X
Minimizing the setup cost of facilities to locations		X	X						X
Minimizing the removal cost of facilities in locations		X							
Minimizing the resources relocation (orientation)	X	X							

2.3.2.1. Frequency of trips between facilities

The expected frequency of trips between facilities made by work personnel is significant because it represents the interaction relationship between any pairs of temporary facilities during each construction phase. Many researchers have considered this factor in their objective function for site layout planning problem when researching their optimization methods. Li and Love (1998) considered the frequency of trips in formulating objective function of site layout planning problem. Lien and Cheng (2012) also included this factor in the formulation. El-Gafy et al. (2010) used this factor in the objective function while applying SA method to solve site layout planning problem. This

factor can be measured by an expected number of trips between facilities per period of interest, such as day or week.

2.3.2.2. Distances between available locations on job site

The travel distances between any pair of available locations on the job site are measured using the travel path of any material movement. While the objective of optimization methods in this problem is to minimize the total traveling distance of site personnel between facilities in terms of total cost, travel distances between locations is one of the important factors in problem formulation. Li and Love (1998) considered this factor in the objective function as well as Lien and Cheng (2012). Andayesh and Sadeghpour (2011) applied MTPE principle to solve site layout planning problem with travel distance between locations factor in the objective function. Although most of researchers use this factor in their problem formulation, they measured the distance between locations differently. Lien and Cheng (2012) measured the distance as rectangular distance between center points of the locations while Mawdesley et al. (2002) measured the distance between two closet points of two facilities by calculating from their coordinates.

2.3.2.3. Material transportation cost between facilities

Transportation cost represents the cost of moving construction materials between facilities within the construction site. Since the facilities are represented in rectangular form, the material transportation cost between facilities represents the cost per distance

unit for each type of material used in the site. Cheung et al. (2002) applied this factor in the objective function when using Evolver^{TM} to solve site layout planning problem. Mawdesley et al. (2002) also considered this factor when determining the fitness for the site layout problem.

2.3.2.4. Shifting cost of facilities between locations

At each phase of the project (except for phase 1) some of the facilities may be already set-up in a specific location within the construction site. Depending on the demands for a specific phase, changes to the location of such facilities may be needed. The shifting cost represents the cost of removing and transporting the facility from its previous to the new location within a facility. The further the facility is moved, the higher the shifting cost; as such, this factor is related to the travel distances between available locations on the job site. Mawdesley et al. (2002) considered this factor in the objective function, but the demonstration of how shifting cost is calculated was not considered. Chandratre and Nandurkar (2011) used this factor GA to solve dynamic layout problem. Shifting cost was considered in their formulation but not validated in their case study. In order to solve dynamic layout problem, El-Rayes and Said (2009) considered relocation cost. The authors considered relocation as moving the facility from its previous location to a different one; this could also be considered as shifting facilities.

2.3.2.5. Setup cost of facilities at any locations

The setup cost is considered at the initial phase of a construction project when all necessary facilities need to be setup for the first time. Afterwards, this factor is considered only if the facilities are required to be moved to a different location or new facilities is coming into site. The setup cost depends on the size of facilities. Mawdesley et al. (2002) mentioned this factor in objective function for site layout problem, but there was no evidence of how the factor can be applied to practical case study.

No weights will be assigned to these requirements. These requirements were chosen based on extensive review and capability of single objective GA. Since the goal of GA in this research is to minimize the total cost of site layout planning, most of cost related factors are considered to make the problem more practical.

2.4. Tracking Technologies

This section describes the key enabling technologies used in cyber-adaptive physical systems and other integration approaches in the construction industry. This section also discusses the role of each technology in enhancing the integration between virtual models and the physical construction. The technologies include barcodes, Wireless sensors and RFID, Cameras, Digital Scanners, Photogrammetry, GPS, Mobile Devices and Communication network.

2.4.1. Radio Frequency Identification Real-time Location Sensing

Many researches have demonstrated efficient ways of implementing RFID technology into construction management by monitoring tagged components/materials on job site. Performance of RFID was investigated for identification during material delivery on job site by (Goodrum et al. 2006; Yagi et al. 2005) and (Song and Mitchell 2006). These approaches required manually scanning tagged components using handheld devices. Ergen and Akinci (2007) further improved on previous approach by developing a prototype for tracking precast materials in the mapped locations of storage yard using RFID and GPS technology. The development demonstrated the potential of RFID system of tracking precast components without manually input status information in the tags. However, GPS feature limits outdoor to indoor usage of the system since component installation status tracking would be embedded. Unlike RFID system, RFID-RTLS system has great potential for both outdoor and indoor usage of tracking tagged components. Over the years, researchers have investigated the use of Radio Frequency Identification Real-time Location Sensing (RFID-RTLS system) for indoor and outdoor construction component tracking applications (Akanmu et al. 2012). The ability to zone or map defined areas of interest is one of the important features of this technology. By fully implementing this feature, the system is able to track tagged components within mapped zone.

2.4.2. Barcode

For long time, barcodes have been used to track materials not only in the construction industry but also in other industries. The use of barcodes was first examined by the beginning for the 1990s. Despite of affordable technology and user-friendly ability, barcodes used in construction industry suffers the problem of having short real range and durability: barcodes require a line of sight and become unreadable if they are scratched or dirty.

2.4.3. 3D Laser Scanning

A laser scanner is a device that analyzes a real-world object or environment to collect data, based on its shape and appearance, which can then be used to construct digital, 3D models useful for a wide variety of applications. 3D laser scanners enable the acquisition of three dimensional as-built information in the form of dense range point clouds or laser scans. At different time on construction site, the laser scanner can be used to generate data. This data can then be used to estimate the quantities of work performed within the time interval considered between two successive scans (El-Omari and Moselhi 2009). Laser scanners have been used for construction quality control (Akinci et al. 2006; Jaselskis et al. 2006), condition assessment (Gordon and Lichti 2004), component tracking (Teizer et al. 2005) and progress monitoring (Bosche and Haas 2008; El-Omari and Moselhi 2009; Su et al. 2006). Although laser scanners are promising in automating data acquisition, there are still a number of challenges in implementing such technology on construction sites. These challenges include mixed pixel phenomenon and

discontinuity of the spatial information as well as scanning range and sensor calibration (Kiziltas et al. 2008). For example, the laser scanner cannot capture the point cloud of a moving object in its line of sight. In addition, the farther the laser scanner is from the objects, the less the level of detail within the captured components. Another potential disadvantage of laser scanning technology is that the operating field conditions of many construction sites, including dirt, extremes of temperature and humidity must be considered because of the sensitivity of laser scanning equipment (Dickinson et al. 2001).

2.4.4. Photogrammetry

Photogrammetry techniques have long assisted in realizing computer aided design (CAD) and virtual models of existing structures for architectural purposes (Dickinson et al. 2001). Photogrammetry is a three dimensional coordinate measuring technology that uses photographs as the medium for measurement. Photogrammetry uses two dimension images and basic triangulation principle to compute the locations of points in three dimensions. The cost and availability of digital cameras provides a good reason to revisit the use of photogrammetry-based approach for as-built documentation. Nowadays, cameras are cheaper, potable and faster for capturing images on site compared to other similar data acquisition technologies. The captured images are later processed using software, to get useful geometric data. However, it is still very difficult to get computers to recognize general objects in images except for structured scenes with severe limits on objects so humans remain an integral part of the photogrammetric path to as-built models (Dickinson et al. 2001). Also, unlike the laser scanning approach, photogrammetry approaches rely on good images so lighting is an issue.

2.4.5. Photography

For many years, photography has proved to be a very useful means of capturing site progress and activities (Brilakis and Soibelman 2005). With the recent advances in digital photography, this method of information gathering is cost-effective, and practical. Owners and contractors can easily acquire construction site photographs from a fixed location to view an ongoing activity or process. Photographs can be taken at different times concurrently in an ongoing construction project and compared for progress monitoring. Previous research efforts in using photographs for the purpose of progress tracking can be traced back to Oglesby et al. (1989) who reported that site photographs allow analysts to focus on the details of the work face while being away from site tensions and confusions, and to perform time-studies on time-lapsed photographs for productivity improvement. However, the lack of advanced technologies for automation had made the process time consuming and unattractive to some extent. More recently, Abeid et al. (2003) presented Photo-Net II wherein time-lapse digital movies of construction activities were linked with critical path activities. In Photo-Net II, time-lapse photography was used as a source of spatial as-built information. In addition, Fard et al. (2007) also recently presented an Augmented Reality (AR) system wherein 3D models are superimposed over time-lapsed photographs. Despite the advantages of photographs, severe weather, illumination and shadow conditions inhibit the utilization of photographs for construction progress monitoring.

2.4.6. Global Positioning Systems (GPS)

GPS is an outdoor satellite-based worldwide radio-navigation system formed by a constellation of 24 satellites, ground control stations, and end users. GPS uses triangulation from these satellites in order to determine a three-dimensional position. To triangulate, a GPS receiver computes the distances to at least four different satellites at any given time. Distances to the receiver are computed measuring the travel time of radio signals from each one of these four satellites. Atmospheric conditions and satellites' location above the receiver influence the resulting position and its accuracy. By default, any computed position has an intrinsic error of about 15 m raw data (Garmin 2004). Differential GPS (DGPS) systems can decrease this error using a fixed receiver whose position must previously be known. There are examples of differential correctors for a GPS system such as: Ground stations, Coast Guard beacons, Wide Area Augmentation System (WAAS), OmniSTAR, and real-time kinematic. These technologies can be summarized as follows:

- Ground stations are fixed GPS receivers that allow post processing the raw data acquired by mobile GPS receivers in a matter of hours. Their resulting accuracy is better than 1 m. The presence of ground based stations close to the receiver is required for this DGPS;
- The U.S. Coast Guard operates Coast Guard Maritime DGPS, another type of fixed ground receiver beacons that enables real-time differential correction U.S. Coast Guard 2004. The accuracy is always better than 10 m, being usually between 1 and 3 m.

- WAAS is a combination of satellites and ground stations that enables real-time differential correction. The resulting accuracy using WAAS satellites are better than 3 m (Garmin 2004);
- OmniSTAR's satellites and network control stations facilitate real-time differential correction of the raw data with worldwide coverage (OmniSTAR 2004). OmniSTAR offers two levels of DGPS. OmniSTAR VBS is a "sub-meter" service, while OmniSTAR HP provides for a position error better than 10 cm. OmniSTAR requires a license fee to use its differential correction; and
- Real-time kinematic GPS (RTK GPS) minimizes the position's error to less than 1 cm in usually less than a 20 mi radius around a separate base station, having a trade-off in highly increased costs.

GPS is an established location technology that offers a wide range of off-the-shelf positioning solutions for the construction industry. Guidance from one location to another (navigation), monitoring the movement of people or assets tracking, creating maps (mapping), and precise timing (timing) are the other GPS basic functions. Although GPS applications are common in construction such as automation of processes and guidance of equipment, the potential of GPS to improve the management of materials on construction job sites remains basically unexplored.

2.5. Building Information Model (BIM)

For many years, building information has been represented via two-dimensional drawing (plan view, section, elevation, etc.). Improvements on the two-dimensional

drawings came in the form of computer aided design (CAD) models. The need for an integrated and collaborative platform for building information delivery gave rise to the development of Building information Models (BIM). Building Information Modeling (BIM) is primarily three dimensional digital representations of a building and its intrinsic characteristics (Hergunsel Sept 2011). Any data attributes and parametric rules for each construction object can be input and stored within BIM as intelligent building components. For example, a window of certain material and dimension is parametrically related and hosted by a wall. Furthermore, BIM provide consistent and coordinated representations of the digital building model including reliable data for each view. BIM also contains project schedule and cost information of construction projects as fourth and fifth dimension. According to the National BIM Standard, BIM is "a digital representation of physical and functional characteristics of a facility and a shared knowledge resource for information about a facility forming a reliable basis for decision during its life-cycle; defined as existing form earliest conception to demolition" ("About the National BIM Standard-United States", 2010). During the design phase of a project, the use of BIM can maximize its impact on a project when the ability to influence cost is the highest. It can be easily realized through the coordination of the project team using a shared model. With this collaborative platform, the project team can jointly visualize and identify issues, provide solutions before they greatly impact projects. Azhar et al. (2008) investigated the use of BIM and pointed out benefits, possible risks for construction industry future research. According to the paper, the key benefits of BIM are listed as follow:

- Fast and effective processes - information is more easily shared, modified and reused
- Better design – building proposal can be easily analyzed, simulation can be quickly performed, and modification can be applied instantly.
- Controlled whole-life cost and environmental data – lifecycle cost are better understood by project team, especially the owner.
- Better production quality – documentation output is flexible and automation feature enhance graphical representation.
- Automated assembly - digital product data can be exploited in real time as construction process starts.
- Lifecycle data – almost all construction related data can be used in facilities management.

According to Stanford University Center for Integrated Facilities Engineering (CIFE), 32 projects using BIM indicate 40% elimination of unbudgeted change, cost estimation accuracy increase 3%, cost estimation process took 80% faster, and 7% reduction in project time. The integration of BIM with high collaboration in project team can not only save time and cost, but also give a great boost in the process of the project. The potential of BIM is much beyond what has been discussed. Having discussed the static aspect of BIM, BIM can be programmed to enhance, improve and evoke its dynamic features or capability. This has tremendous opportunities for providing access to real-time project monitoring and control information. This research will implement the integration of BIM with RFID-RTLS system for automated status tracking of tagged components, specifically for site layout generation.

2.6. Limitations of Previous Work

Previous studies have demonstrated the potential for significant improvements to project delivery process based on the use of optimization techniques to determine optimal layout. Also, studies have demonstrated the potential of sensing technologies for tracking building components. Opportunities exist for integrating the sensing technologies and optimization techniques for effective tracking of tagged components and real-time generation of site layout based on sensed data. This offers opportunities for reducing the time wasted by the project team when investigating site availability. This will also enhance safety of project team especially for highly congested sites.

CHAPTER 3: RESEARCH METHODS/APPROACH

3.1. Introduction

Based on the requirements identified in the preceding section, this chapter presents the problem formulation. This chapter all describes the GA approach to implementing the formulated problem. The role of the BIM and RFID-RTLS system in developing the proposed automated system is also presented.

3.2. Problem Formulation

Based on the factors identified in section 2.2.2, this section introduces the objective function.

3.2.1. Objective Function

The objective of this site layout problem is to minimize the total layout cost. The total site layout cost is defined as in Equation (3.1):

Minimize

$$\sum_{i=1}^{M}\sum_{k=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{N} F_{ik}d_{jl}\,C_{ik}X_{ij}X_{kl} + \sum_{i=1}^{M}\sum_{j=1}^{N} S_iX_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{ijl}d_{jl}Y_{ijl} \quad (3.1)$$

Where i and k are the i^{th} and k^{th} facilities; j and l are the j^{th} and l^{th} locations; M is the total number of possible facilities to install; N is the total number of locations

available; F_{ik} is the frequency of trips matrix between facilities i and k; d_{jl} is the distance between location j and l; C_{ik} is the material transportation cost between facilities i and k; A_{ijl} is the shifting cost of facility i between location j and l; S_i is the setup cost of facility i; X_{ij}, X_{kl} is the decision variable that indicates if facility i, k is installed in location j, l; Y_{ijl} is also decision variable that indicates any moving facilities at current phase.

Subject to, $\sum_{i=1}^{M} X_{ij} \leq 1$, only 1 facility can be assigned to 1 location. (3.2)

$\sum_{j=1}^{M} X_{kl} \leq 1$, only 1 location can be assigned to 1 facility. (3.3)

$X, Y \in \{0, 1\}$ (3.4)

Equation (3.2), (3.3) and (3.4) are the constraints of objective function (3.1) where (3.2) and (3.3) only allow one facility to be assigned to one selected location and vice versa. Equation (3.4) sets the permutation matrices X and Y to binary numbers; indicating the facilities that belong to locations. Note that the first term of the objective function contains the product of two decision variables, which makes the problem a non-linear optimization problem. However, by using exact methods and solve with GA, the problem will be treated as a linear optimization problem.

3.3. Genetic Algorithm

Genetic Algorithm (GA) is a search optimization inspired by the process of natural evolution in Biology (Goldberg 1989). GA belongs to a larger class called Evolutionary Algorithm (EA), which mimics the similar concepts by using evolution operators such as mutation, crossover, selection or inheritance. These operators are

explained in detail in the next section. In this research, genetic algorithm is examined as a possible method to solve site layout planning problem. GA is very suitable for permutation optimization problem such as site layout planning because of its flexible performance and coding ability under many types of fitness function (discrete, linear or non-linear).

GA deploys a population of linear permutation encoded solutions, known as chromosome. These solutions can be coded as binary (0s and 1s), integer, real number. Each chromosome (possible solution) will then be evaluated by objective function (minimization or maximization) as well as after every generation. New or current population of potential solution will be kept for the next generation in order to produce better result. Between iterations, genetic operators are applied to current population of layout solution in order to produce new ones with potentially better result. First, selection operator is applied to pair any chromosome in population together to form parents. Second, crossover operator will randomize elements (facilities) between parents to produce better combination. Lastly, mutation operator is used to maintain the genetic diversity of the population from one generation to the next. At the end of iterations, better fitness of objective function would be produced. The algorithm is only allowed to stop once stopping criteria is met; for instance, a maximum iteration or a satisfied fitness occurred. GA will then be implemented in MATLAB (the sample codes is presented in the appendix) and validated in chapter 5 using a medium-sized project previously developed by Li and Love (2000) as a hypothetical construction site. Lien and Cheng (2012) also used this project as a case study to determine the optimum site layout using

the proposed hybrid swarm algorithm. This research validates the results of the proposed GA approach with that generated by Lien and Cheng (2012).

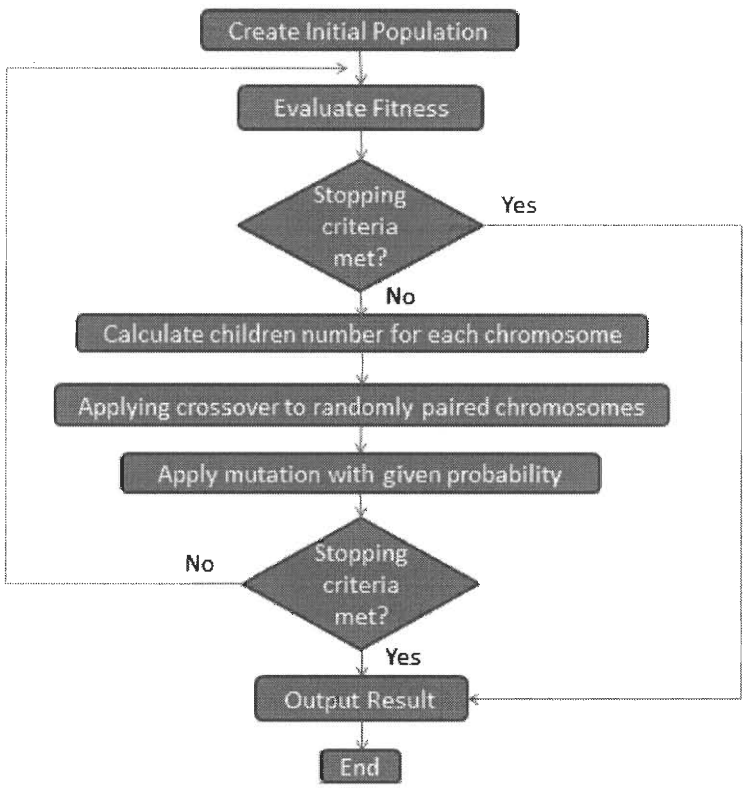


Figure 3-1 - Genetic algorithm flowchart

3.3.1. Encoding

In order to GA to understand location layout and list of temporary facilities, permutation encoding is applied to this problem. There are two ways to present this issue such as: string layout representation and permutation matrix. First, string layout representation is to show the users which facilities are assigned to which locations. Second, permutation matrix is treated as decision table of all zeros, but wherever facility is assigned to a location, it will be assigned as 1. There is only number 1 in each row and

each column. Permutation matrix is used in objective function as $X_{ij}X_{kl}$. Although permutation matrix and string layout representation is different, they both carry the same purpose of presenting the solution layout.

Table 3-1 – String layout representation

Facilities	9	3	7	10	11	6	2	5	8	1	4
Locations	1	2	3	4	5	6	7	8	9	10	11

Table 3-2– Permutation matrix with 11 facilities

		Facilities										
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Locations	**1**	0	0	0	0	0	0	0	0	1	0	0
	2	0	0	1	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	1	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	1	0
	5	0	0	0	0	0	0	0	0	0	0	1
	6	0	0	0	0	0	1	0	0	0	0	0
	7	0	1	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	1	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	1	0	0	0
	10	1	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	1	0	0	0	0	0	0	0

3.3.2. Selection

Selection (or reproduction) is the first operator to be applied in a genetic algorithm implementation. It consists of selecting two chromosomes (parents) from a given population. The selected chromosomes will be used in the operators (such as crossover and mutation). There are several types of selection approach, including:

roulette wheel selection, Boltzmann selection, tournament selection, rank selection and steady state selection.

Roulette wheel selection, also known, as Fitness Proportionate selection is one of the most commonly used selection operator. The concept can be thought as a game of roulette; in which parents are selected based on their fitness. The better the fitness of the chromosomes is, the higher the chances to be selected. In some cases, there will be one chromosome with a high fitness (e.g. 90%) that dominates others. If dominance exists, then the population's genetic diversity is likely low and it will take more iteration to produce near optimal solution. Rank selection assigned a numerical rank to each chromosome based on their fitness and thus prevents very fit individuals from gaining dominance early. In this particular problem, rank selection operator is used.

3.3.3. Crossover

The crossover operator is also known as genetic operator (or mating process) between two selected chromosomes to produce offspring (new possible solution). The ultimate goal of genetic operator is to produce better chromosomes than both parents. There are many available crossover operators such as: one-point, two-point, uniform, cycle, arithmetic and heuristic crossover. The most appropriate crossover operators depend on the coding of chromosomes. For optimization problems that involve permutation, cycle crossover (CX) is the most suitable for generating offspring between parents. In CX, an index of facilities string (one of the chromosome of the population) is randomly selected where the integers are exchanged between parents. If the exchanged

integers are not the same, each offspring has a duplicate integer. Then, the repeated integer is switched between 2 offspring. Now, a different integer is duplicated; therefore, the first exchanged site is returned as iteration goes on. By now, each offspring should contain exactly one copy of each integer. There is probability of crossover operator that represents how fast the new solutions can be introduced into the population. The adaptive range of crossover probability is 0.5 – 1.0 (Srinivas and Patnaik, 1994).

Procedure of cycle crossover is explained as follow:

Chromosome 1:

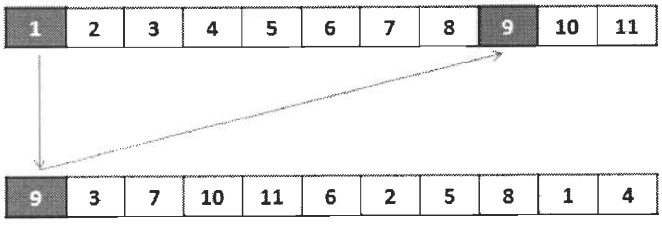
1	2	3	4	5	6	7	8	9	10	11

Chromosome 2:

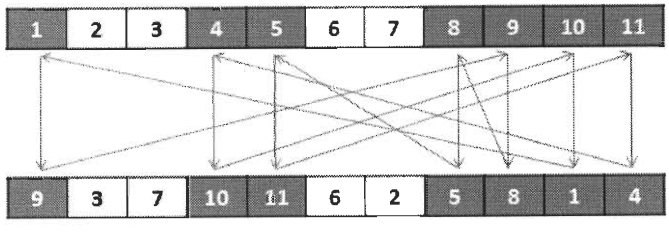
9	3	7	10	11	6	2	5	8	1	4

Step 1 – selected at the first allele of chromosome 1 (first parent)

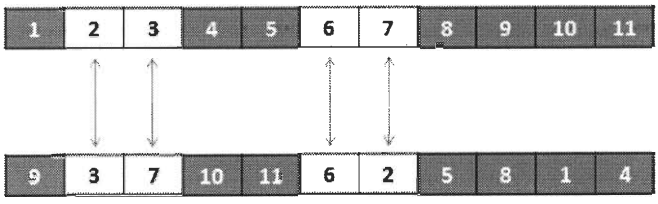
Step 2 – Look up the same position of that allele in chromosome 2 (second parent)



Step 3 – Repeat step 1 and 2 until it returns to the first allele of chromosome 1



Step 4 – Exchange position of all remaining alleles between 2 chromosomes



Step 5 – Produce offspring

Offspring 1:

1	3	7	4	5	6	2	8	9	10	11

Offspring 2:

9	2	3	10	11	6	7	5	8	1	4

3.3.4. Mutation

Mutation is the last and most important operator in genetic algorithm as it helps prevent the population from stagnating at local optima. The purpose of mutation is to maintain the genetic diversity of the population from one generation to the next. Depending on user-defined mutation probability, the chromosome may change completely from the previous generation. Caution is required when setting the mutation parameter since a high probability of mutation could result in a primitive random search while a low probability of mutation limits the potential of exploration. The preferred range of mutation rate is usually from 0.5% to 1%. In this problem, the mutation operator will randomly choose 2 facilities in a chromosome and exchange their positions.

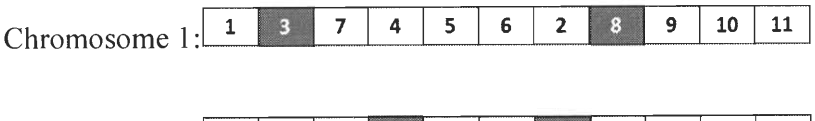
Chromosome 1:

1	3	7	4	5	6	2	8	9	10	11

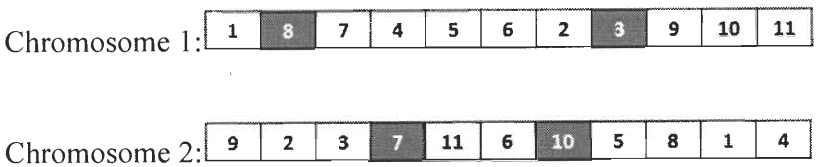
Chromosome 2:

9	2	3	10	11	6	7	5	8	1	4

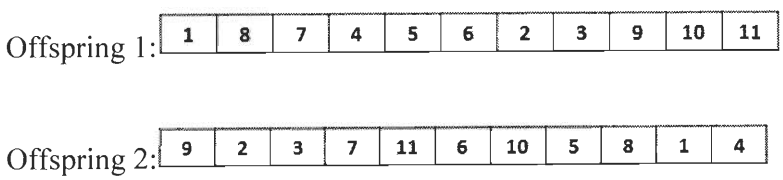
Step 1- Randomly pick any two alleles in a chromosome



Step 2 – Swap their positions within a chromosome



Step 3 – Produce offspring



3.4. Construction Site Mapping using RFID-RTLS System

The advantage of GA (as highlighted in Section 2.2.1.9) is that it needs no determination of problem space. This means that as long as the number of locations and temporary facilities are defined, GA will be able to generate site layout. This flexibility is great for the implementation with RFID-RTLS system. The RFID-RTLS system has four main components as shown in Figure 3-2 namely master reader, slave readers, RTLS tags and location engine software. The RTLS tags are battery powered and typically attached to assets and personnel to be tracked. The tags relay location information every

predetermined time interval (also known as cycle time) to the master and slave reader. The readers are positioned at the boundaries of areas of interest or observation areas. The slave readers determine the range of the tags and relay the data to the master reader. In addition to determining the range of the tags, the master reader also captures the data from each slave reader and relays it to the location engine software (installed in the associated computer system). The location engine software determines the location of the RTLS tags based on time-of-arrival (TOA) of radio frequency signal from the tags to the reader, where the distance between the tags and reader (the master and slave readers are positioned at known reference locations) are determined by the round-trip time of the signal travelling between the two entities.

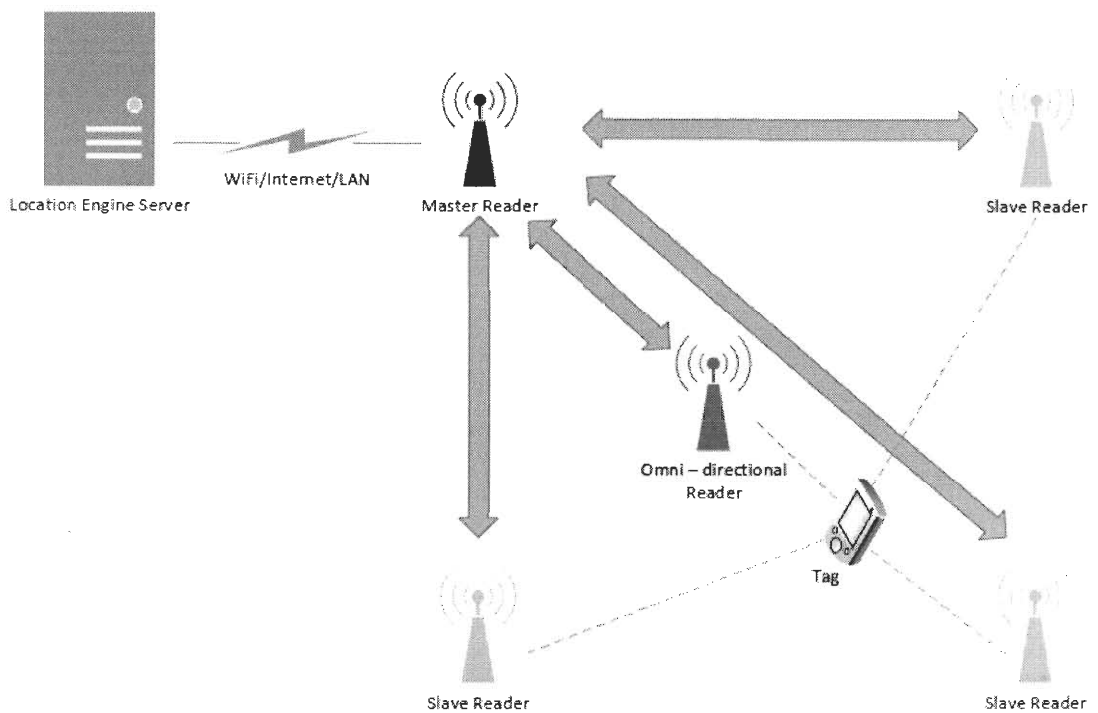


Figure 3-2 - Component overview of the RFID-RTLS system

In the RFID-RTLS system, areas of interest can be graphically mapped out. This research will take advantage of this spatial mapping feature of the RFID-RTLS system to

create an automated system. Construction manager will input information about area of interest into RFID-RTLS system. GA in MATLAB can create the site layout and display on the RFID-RTLS system interface. By defining area of interest, the RFID-RTLS system can calculate coordinates of possible locations, restricted areas, etc. Furthermore, the RTLS system can be integrated with virtual models such as BIM: the resulting system will not only enable real-time visualization of status of tagged component e.g. from entry on-site to installation (Akanmu et al. 2012), but also track project schedule.

3.5. Project Schedule Tracking using BIM

The project schedule is an important feature in BIM (discussed in section 2.4). The project schedule comprises of the activities or tasks and their duration. The BIM software used for this research is Autodesk Navisworks. Navisworks was utilized because it offers a .Net application-programming interface (API), which enables users to write custom plug-ins to drive Autodesk Navisworks from outside the graphical user interface (GUI) and automate tasks like timeline properties. Using the API of the BIM software and Visual Studio .Net, the tasks and durations within BIM can be programmed to automatically generate a site layout. In this research, construction managers will input project schedule into BIM and the add-in feature will real time monitor the process. The project schedule will be divided into three main phases: substructure, superstructure and interior finish. At the end of each phase, site layout generation will be activated. If construction managers decide to end any construction phases early, the program will understand this and generate the site layout accordingly.

CHAPTER 4: PROTOTYPE SYSTEM DEVELOPMENT

4.1. Introduction

This chapter describes the development of two prototype systems based on the research approach described in the preceding chapter. The system and software architecture for both systems are presented. The chapter also presents use case diagrams to illustrate the usability of the developed prototypes.

4.2. Prototype #1: Site Layout Planning using Genetic Algorithm

For modeling the construction site layout problem, the following factors were considered: expected frequency of trips between facilities, distances between available locations on job site, expected material transportation cost between facilities, shifting cost of facilities between locations and setup cost of facilities at any location. These are briefly described below:

4.2.1. Frequency of Trips between Facilities

The expected frequency of trips between facilities made by work personnel (Li and Love 2000) represents the interaction relationship between any pairs of temporary facilities during each construction phase. This factor is measured by an expected number of trips between facilities per period of interest, such as days or weeks. This factor has earlier been used by assumption (Li and Love 2000) as shown in Table 5-2.

4.2.2. Distances between Available Locations on Job Site

The travel distances between any pair of available locations on the job site are measured using the travel path any material movement. Table 5-3 shows the travel distance between facilities as defined by (Li and Love 2000). This is generally the rectangular distance between the center points of the locations

4.2.3. Objective Function

The objective function is similar to equation (3.1) in previous section 3.1.1. Since this system is validated by comparing results with Lien and Cheng (2012) and Li and Love (2000), the same factors considered by the authors will be used in the proposed GA. The factors include travel distance and frequency of trips between facilities.

Where i, k is i^{th} and k^{th} facilities; j, l is j^{th} and l^{th} locations; M, N is the number of facilities and locations, respectively; F_{ik} is the frequency matrix between facilities i and k; d_{jl} is the distance matrix between location j and l; X_{ij}, X_{kl} is the decision variable that indicates if facility i, j is installed in location k, l.

Minimize $$\sum_{i=1}^{M}\sum_{k=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{N} F_{ik} d_{jl} X_{ij} X_{kl} \quad (4.1)$$

Subject to $$\sum_{i=1}^{M} X_{ij} \leq 1, \text{ only 1 facility can be assigned to 1 location.} \quad (4.2)$$

$$\sum_{j=1}^{M} X_{kl} \leq 1, \text{ only 1 location can be assigned to 1 facility.} \quad (4.3)$$

$$X, Y \in \{0, 1\} \quad (4.4)$$

A Matlab code was developed based on the objective function presented above. The following section presents the use-case analysis of the developed GA approach.

4.2.4. Use Case Analysis

This use-case (Figure 4-1) describes how the prototype works and what functions a construction manager can perform within the system. GA as mentioned previously is implemented in the system, and the use-case also describes how a manager will use it to achieve cost effective site layout solution.

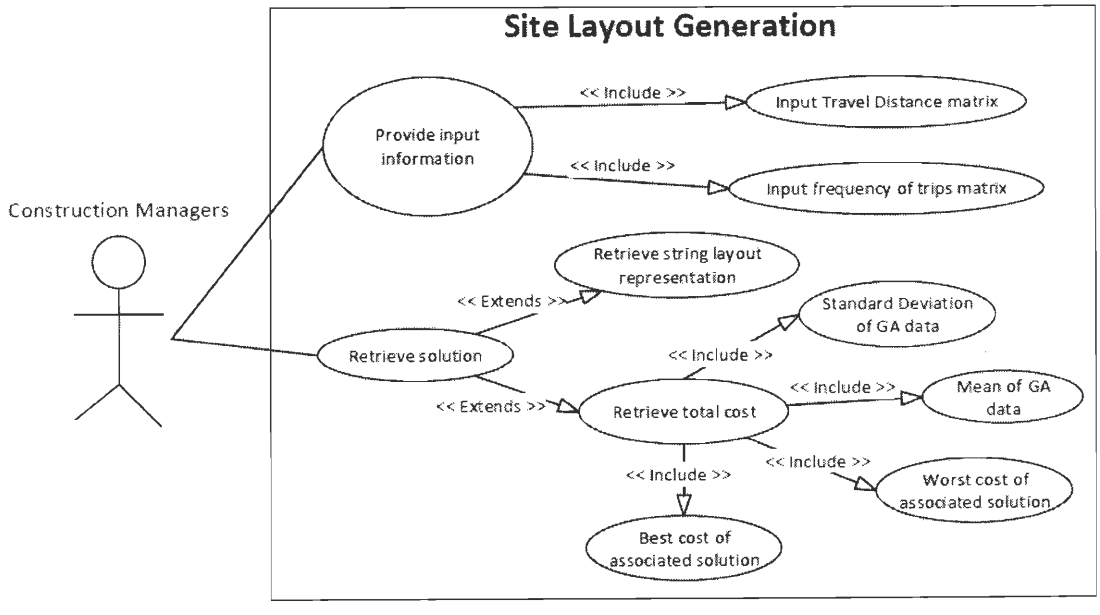


Figure 4-1– Prototype #1 use case analysis

Primary Actor: Construction Manager

Short Description: This system provides a cost effective site layout solution based on the following inputs: frequency of trips made by personnel between temporary facilities and travel distances between possible locations.

Preconditions:

- Number of available locations on site must be known.
- Travel distance between available locations must be known and expressed as matrix in Excel.
- Frequency of trips between temporary facilities needs to be expressed as a matrix in Excel.
- Main file in MATLAB program needs to be opened.

Basic Flow:

- The construction manager inputs travel distance matrix and frequency of trips matrix into Excel.
- Excel files must be in same folder as MATLAB main file.
- The manager opens the main file for site layout planning in MATLAB program.
- The manager clicks the button "Save and Run" in MATLAB program interface to execute the file.
- The duration of execution of the program depends on the number of locations/facilities involved.
- Site layout solution is produced as string layout representation as mentioned in Table 3-1 of section 3.2.1.

- Layout cost is shown along with the layout solution for manager's reference.

Alternative Flows:

- After the 1st phase of construction, if available locations on the site changes and new facilities are needed, the managers must modify matrices in Excel.
- If any temporary facility remains fixed between phases, the manager can remove the locations and facilities in travel distances and frequency of trips matrix, respectively.

Post-conditions:

- Site layout solution is displayed as string layout presentation.
- Layout cost is displayed in MATLAB as well as mean, standard deviation of GA data set.

4.3. Prototype #2: Automated Site Layout Generation

Once prototype #1 is validated, the algorithm is integrated with RFID-RTLS system and BIM to create an automated site layout system (Prototype #2). This is described as follows

4.3.1. System Overview

Prototype system #2 consists of the RFID-RTLS system, Building Information Model (BIM) and Prototype #1 (in the form of a Matlab code) integrated using Visual

Studio .Net. In order for three systems to communicate, all output and input from each system will be transferred into MySQL database.

4.3.2. Key Aspects

The following are the key aspects of the automated site layout generation system:

- Real-time materials/components tracking within spatially mapped zones;
- Providing context of spatially mapped layouts during construction;
- Tracking of the project schedule; The system automatically adjusts as project schedule changes;
- Automated site layout generation after each construction phases;
- In transition of construction phases, the system automatically understands and adjusts input information based on current situation on site and project schedule.

4.3.3. System Architecture

Figure 4-3 illustrates the architecture of the automated site layout system. The system architecture brings together the key enabling technologies described in Chapter 3.

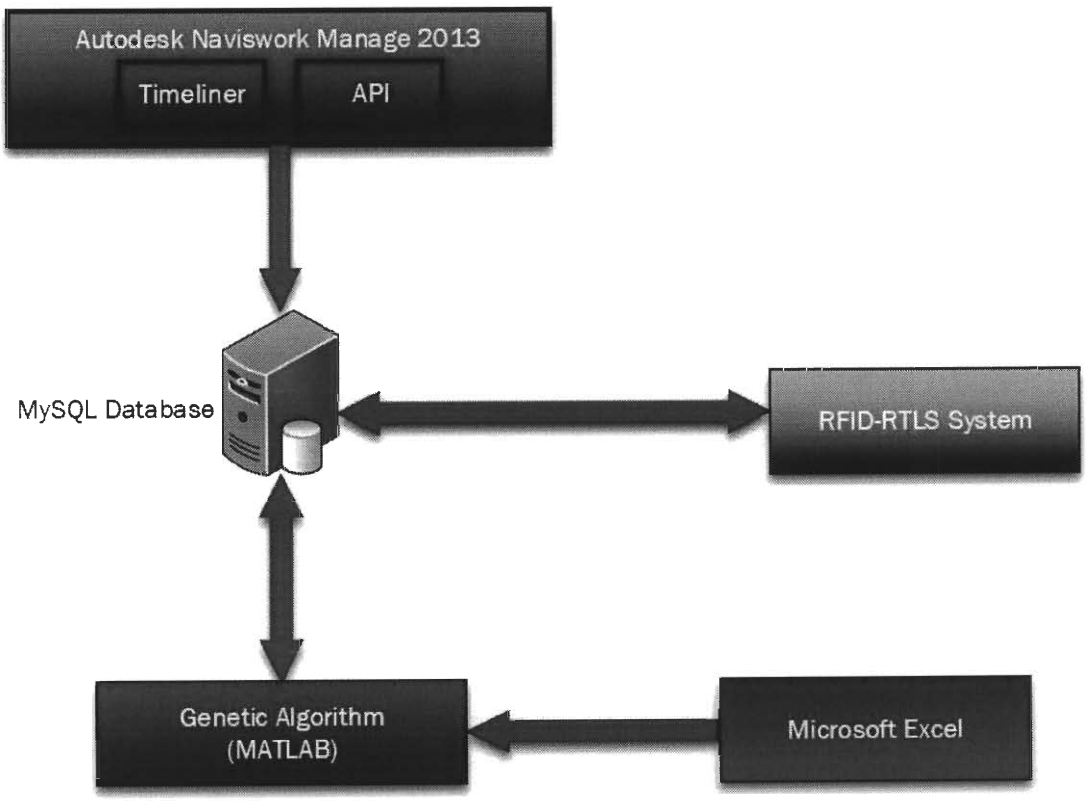


Figure 4-2 – Software architecture

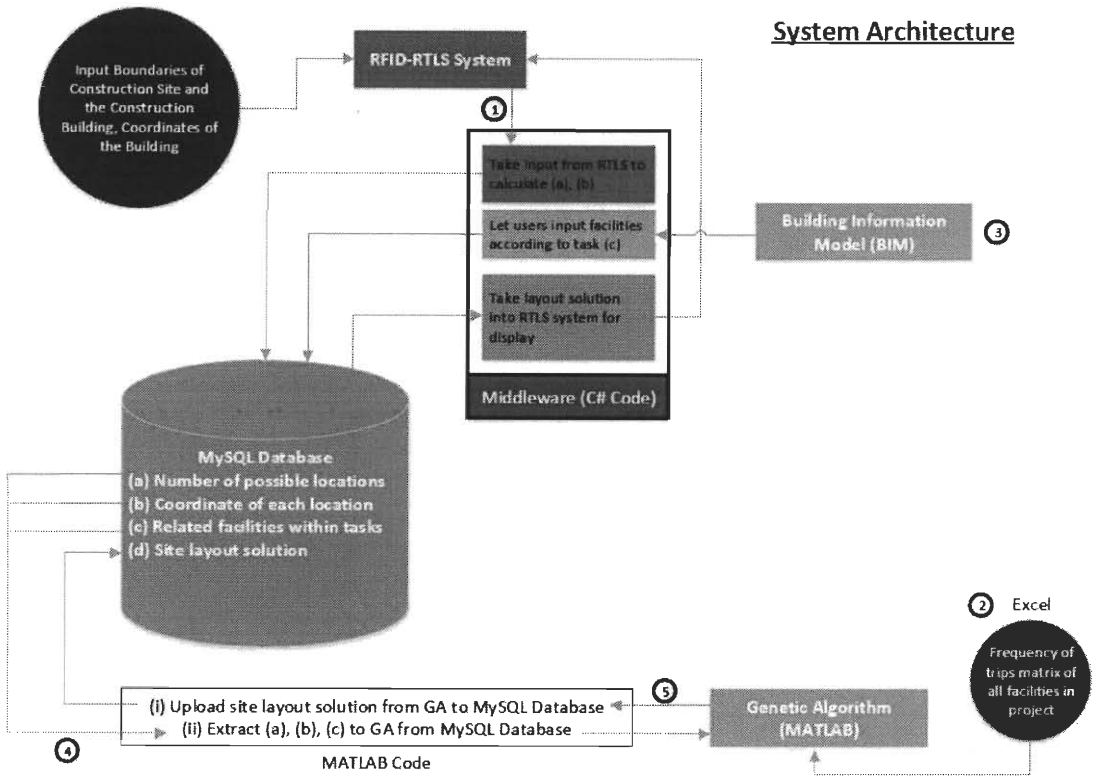


Figure 4-3– System architecture

The site boundary and coordinates are initially prepared and created in the FRID-RTLS system interface. The middleware collects the data from the RFID-RTLS system information, computes the number and coordinates of possible locations and represents them in the MySQL database. The frequency of trips matrix is developed in Excel (show in Figure 4-6) and collected as an input to the GA by the Matlab code. The BIM contains the project schedule, which includes the different phases, related tasks and activities. Within BIM, the middleware enables the user to insert facilities related to each task. The entry interface within the BIM software is shown in Figure 4-5. The middle ware collects the BIM information and enters it into the database. On entry into the database, the database sniffer from the middleware triggers the GA Matlab code to generate the site

layout. The Matlab code in GA uses the information contained in the MySQL database (number and coordinates of possible locations and the list of relate facilities) to generate the site layout. On generating the layout, the Matlab code sends the layout to the MySQL database. The middleware captures the layout and displays it using the RFID-RTLS system visualization interface. The software architecture, which includes the interaction between the different software components, is shown in Figure 4-2.

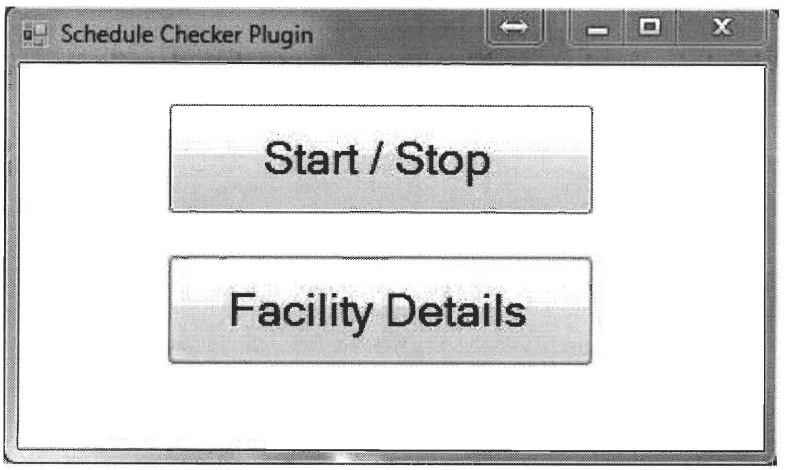


Figure 4-4 – Schedule checker add-in interface

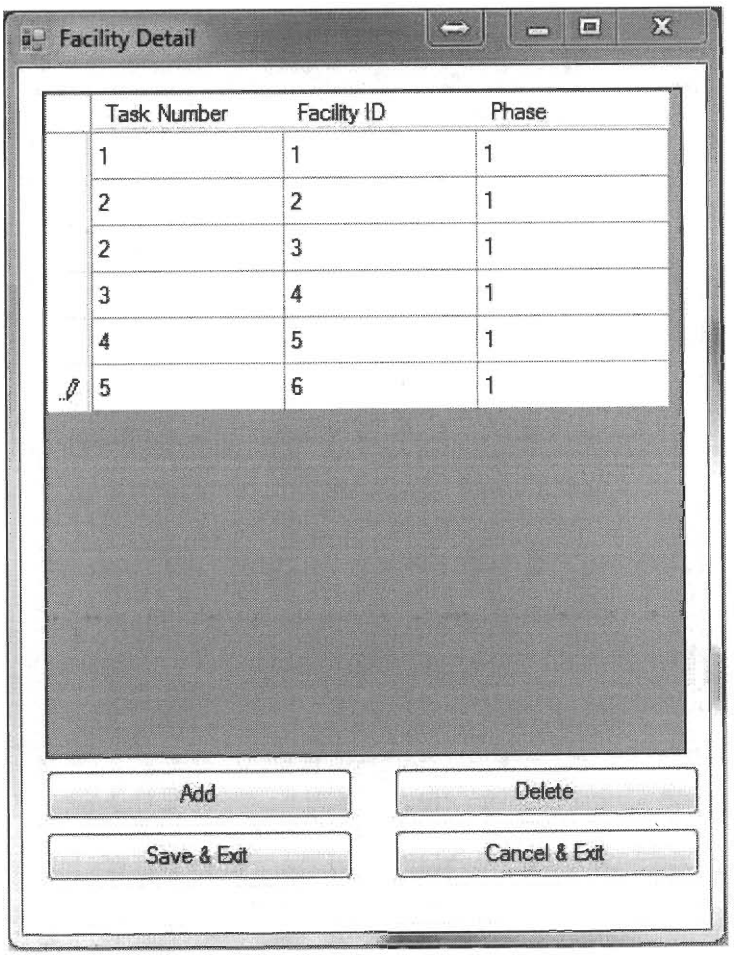
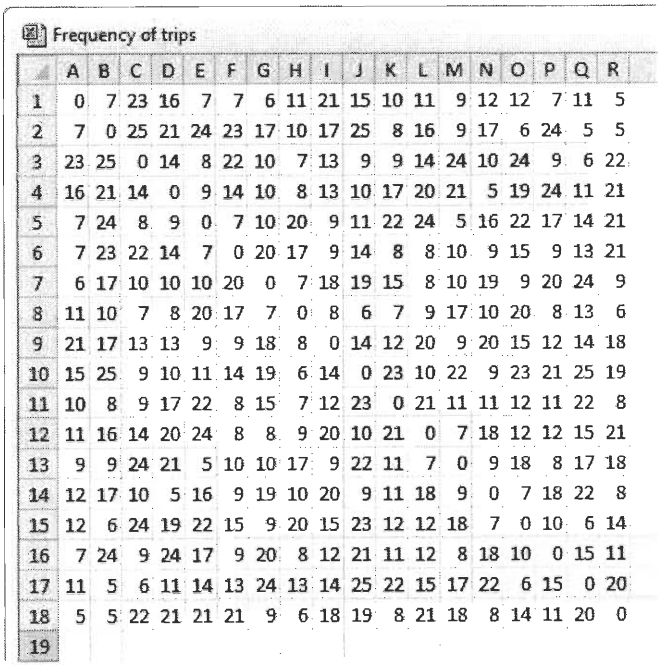


Figure 4-5 – Entry interface for related facilities within BIM



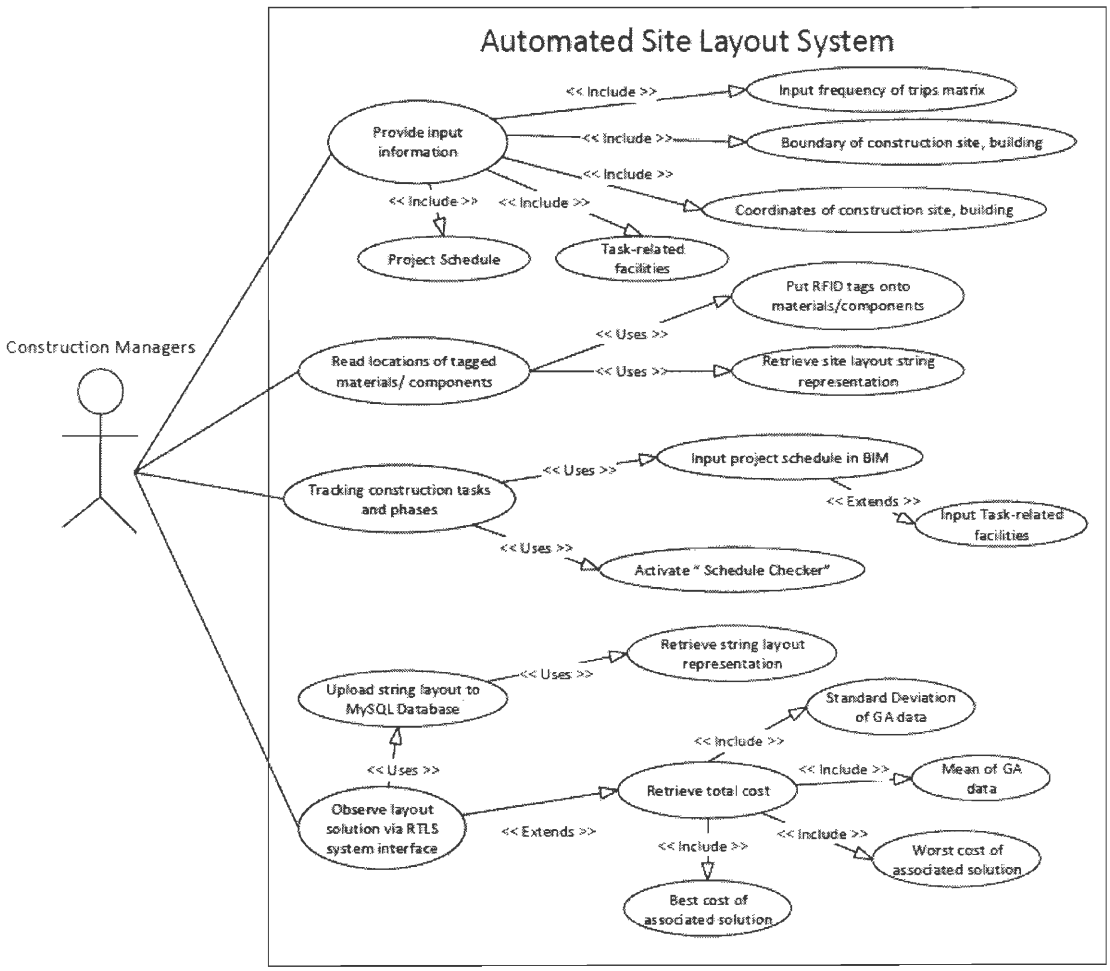
Frequency of trips

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	0	7	23	16	7	7	6	11	21	15	10	11	9	12	12	7	11	5
2	7	0	25	21	24	23	17	10	17	25	8	16	9	17	6	24	5	5
3	23	25	0	14	8	22	10	7	13	9	9	14	24	10	24	9	6	22
4	16	21	14	0	9	14	10	8	13	10	17	20	21	5	19	24	11	21
5	7	24	8	9	0	7	10	20	9	11	22	24	5	16	22	17	14	21
6	7	23	22	14	7	0	20	17	9	14	8	8	10	9	15	9	13	21
7	6	17	10	10	10	20	0	7	18	19	15	8	10	19	9	20	24	9
8	11	10	7	8	20	17	7	0	8	6	7	9	17	10	20	8	13	6
9	21	17	13	13	9	9	18	8	0	14	12	20	9	20	15	12	14	18
10	15	25	9	10	11	14	19	6	14	0	23	10	22	9	23	21	25	19
11	10	8	9	17	22	8	15	7	12	23	0	21	11	11	12	11	22	8
12	11	16	14	20	24	8	8	9	20	10	21	0	7	18	12	12	15	21
13	9	9	24	21	5	10	10	17	9	22	11	7	0	9	18	8	17	18
14	12	17	10	5	16	9	19	10	20	9	11	18	9	0	7	18	22	8
15	12	6	24	19	22	15	9	20	15	23	12	12	18	7	0	10	6	14
16	7	24	9	24	17	9	20	8	12	21	11	12	8	18	10	0	15	11
17	11	5	6	11	14	13	24	13	14	25	22	15	17	22	6	15	0	20
18	5	5	22	21	21	21	9	6	18	19	8	21	18	8	14	11	20	0
19																		

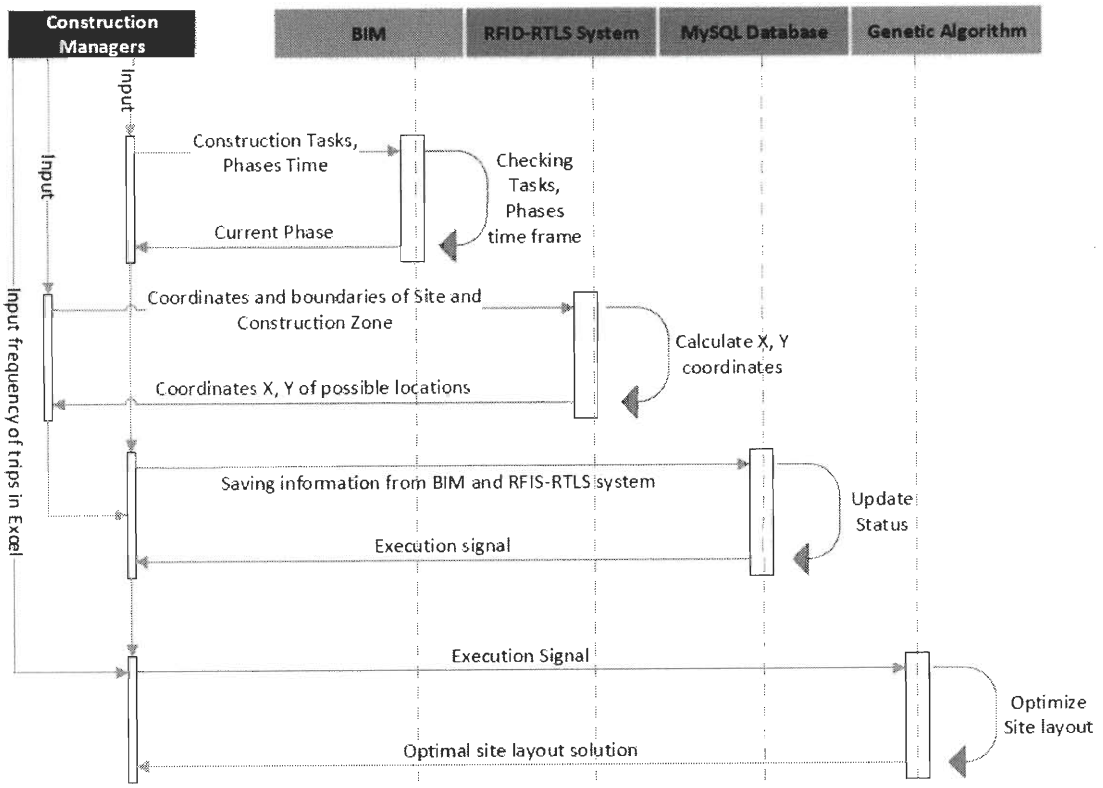
Figure 4-6 – Developed frequency of trips matrix in Excel

4.3.4. Use Case Analysis

This use-case (shown in Figure 4-7 – a) describes how a construction manager would use the system. Figure 4-7 – b is the sequence diagram which explains a step-by-step process of how the automated system executes.


Automated Site Layout System
Construction Managers
Provide input information
<< Include >>
Input frequency of trips matrix
<< Include >>
Boundary of construction site, building
<< Include >>
Coordinates of construction site, building
<< Include >>
Project Schedule
<< Include >>
Task-related facilities
Read locations of tagged materials/ components
<< Uses >>
Put RFID tags onto materials/components
<< Uses >>
Retrieve site layout string representation
Tracking construction tasks and phases
<< Uses >>
Input project schedule in BIM
<< Extends >>
Input Task-related facilities
<< Uses >>
Activate " Schedule Checker"
Upload string layout to MySQL Database
<< Uses >>
Retrieve string layout representation
<< Uses >>
Observe layout solution via RTLS system interface
<< Extends >>
Retrieve total cost
<< Include >>
Standard Deviation of GA data
<< Include >>
Mean of GA data
<< Include >>
Worst cost of associated solution
<< Include >>
Best cost of associated solution

(a)



(b)

Figure 4-7– (a) Prototype #2 use case (b) Sequence diagram of the system

Primary Actor: Construction Managers

Short Description: Site layout solution is automatically generated based on the project schedule, status of tagged components and space availability.

Preconditions:

- Boundary of area of interest needs to be defined
- Area of construction building (if within area of interest) also needs to be defined.

- Coordinates of construction building closet to selected benchmark must be available.
- All materials/components on job site must be tagged.
- Modified RFID-RTLS system must run.
- Project schedule must be known.
- List of facilities involved in construction tasks must be known.
- Navisworks Manage 2013 must be opened.
- Schedule Checker add-on feature must be activated.
- Frequency of trips between temporary facilities must be expressed as matrix in Excel.
- Main file in MATLAB must be opened.

Basic Flow:

- The construction manager inputs frequency of trips matrix into Excel.
- The manager executes the add-in RFID-RTLS system code.
- Boundary of area of interest, construction building and coordinates are input.
- The construction personnel tag the materials/components on job site.
- The manager inputs the project schedule (project schedule must be divided into three main phases) into Navisworks. Note that any overlap task between phases cannot be grouped within their starting phase.
- The manager clicks "Schedule Checker" in Add-ins tab within Navisworks Manage 2013 interface.
- The manager clicks "Facility Involve" in the "Schedule Checker" interface.

- The manager inputs the list of facilities involved in each task. Note that if there is more than 1 facility within a task, the manager needs to duplicate the task and add new facilities.
- The manager clicks "Save & Exit" in "Facility Details" interface as shown in Figure 4-5.
- The manager clicks "Start/Stop" in "Schedule Checker" interface as shown in Figure 4-4.
- The manger clicks "Save and Run" in MATLAB program to execute main file.
- The duration of execution of the program will depend on the number of locations/facilities involved.
- Site layout solution is produced and displayed in RFID-RTLS system interface
- Personnel place tagged materials/components into their assigned places.

Alternative Flows:

- In the 1st phase of construction, if the construction manager ends a task which is supposed to end in phase 2, the system will know that the facilities involved in that task can be moved in the next site layout generation (if the facility is empty). This will also happen if a facility is no longer required; the construction manager needs to update the information in "Facility Details" within Navisworks.

Post-conditions:

- Site layout solutions are displayed in the RFID-RTLS system GUI.
- Movement and locations of tagged materials/components on site can be monitored via RFID-RTLS system interface.
- Layout cost can be displayed in MATLAB.

4.3.5. System Implementation

Figure 4-8 illustrates how the step-by-step process of implementing the automated site layout system. It graphically explains how the system would react at different checkpoints.

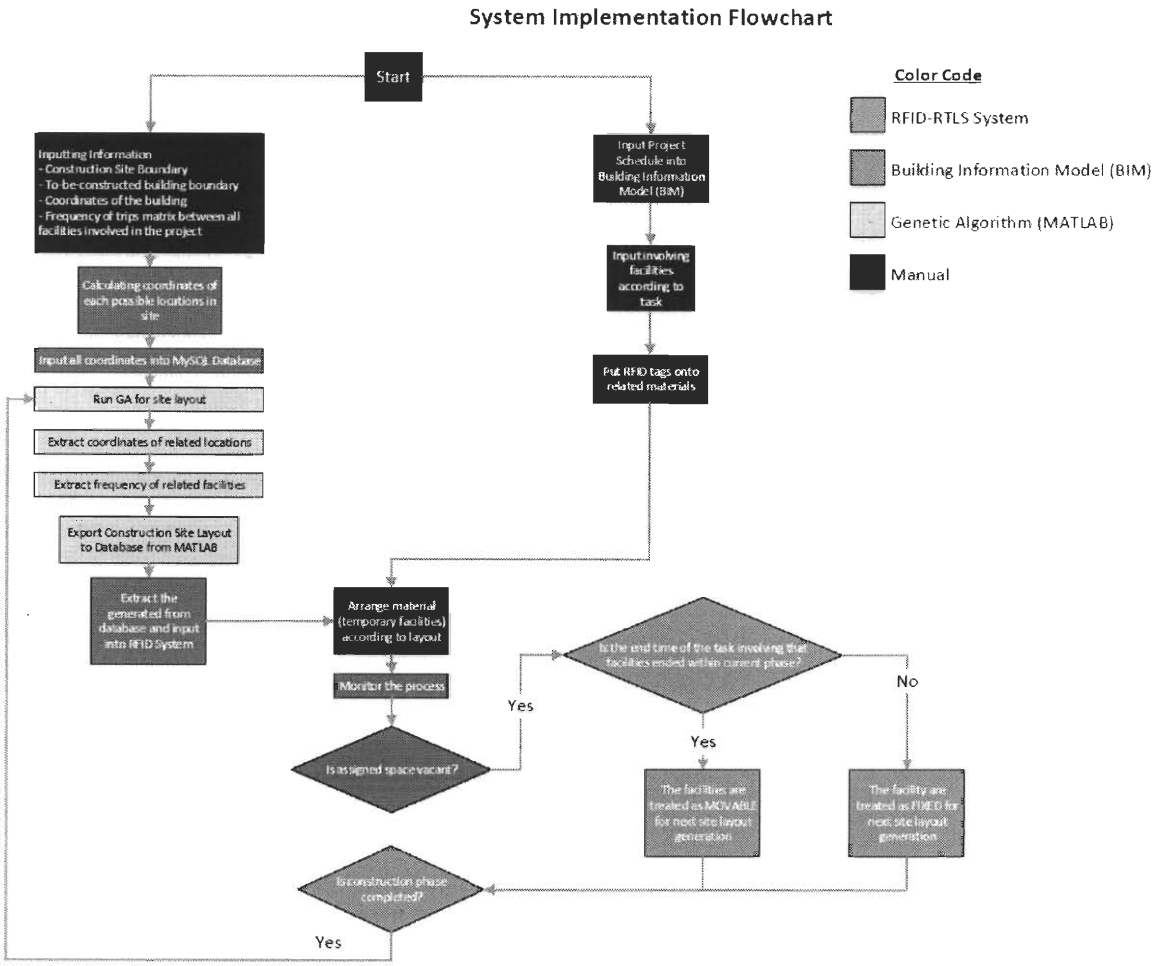


Figure 4-8– System flowchart

Step 1 – The users provides in the RFID-RTLS system and BIM information inputs such as boundary of construction site and construction building, coordinates of the building, frequency of trips matrix between all temporary facilities involved in the project, project schedule and list of involving facilities according to task. Step 2 – The RFID-RTLS system obtains coordinates of each possible location for temporary facilities on site.

Step 3 – All calculated coordinates will be uploaded onto MySQL database.

Step 4 – GA in MATLAB is executed.

Step 5 – GA in MATLAB extracts coordinates of locations according to facilities involved in the current phase from MySQL database and calculates travel distance matrix for use in the objective function.

Step 6 – GA in MATLAB calls out frequency of trips matrix between any facilities involved in the current phase for the objective function.

Step 7 – Export site layout solution to MySQL database.

Step 8 – RFID-RTLS system extracts and display layout solution in the interface.

Step 9 – Personnel tags materials/components according to layout solution.

Step 10 – The construction managers monitor the construction process.

Step 11 – RFID-RTLS system keeps checking if any temporary facilities in the layout are vacant. If there are no vacant facilities, the system does nothing.

Step 12 – If there are vacant facilities, BIM checks to see if the task associated those facilities ends within the current phase.

Step 13 – If the task ends within the current phase, the facilities are registered as "movable" for next site layout generation. If not, the facilities are registered as "fixed".

Step 14 – BIM keeps checking if the construction phase has completed. If the construction phase has completed, the system returns to step 4. If otherwise, the system does nothing.

CHAPTER 5: PROTOTYPE SYSTEM VALIDATION AND CASE-STUDY

5.1. Introduction

This chapter describes the validation and implementation of the developed prototypes on case-study projects on Western Michigan University campus. The chapter also presents and discusses the results obtained from the implementation of the prototypes.

5.2. Prototype #1 Validation

5.2.1. Overview

To validate the proposed GA, this paper used a medium-sized project found by Li and Love (2000) as a hypothetical construction site. Lien and Cheng (2012) also used this project as a case study to determine the optimum site layout using the hybrid swarm. The hybrid swarm algorithm integrates the Particle Swarm optimization (Maier et al. 2003) and the Bee algorithm (BA) to form the Particle Bee algorithm (PBA). This study validates the results of the proposed GA approach with that generated by Lien and Cheng (2012) using the hypothetical construction site. In the study by Li and Love (2000), they regarded the side and main gate facilities as fixed as this can be predetermined by planners. In this study, the locations are replaced with 2 facilities (H and K) which are dumpster and temporary restroom as shown in Table 5-1. A sample layout of the site is shown in Figure 5-1.

Table 5-1– List of temporary facilities for case study

Facility Number	Facility Name
A	Site Office
B	Falsework workshop
C	Labor residence
D	Storeroom 1
E	Storeroom 2
F	Carpentry workshop
G	Reinforcement steel workshop
H	Dumpster
I	Electrical, water and other utilities control room
J	Concrete batch workshop
K	Temporary restroom

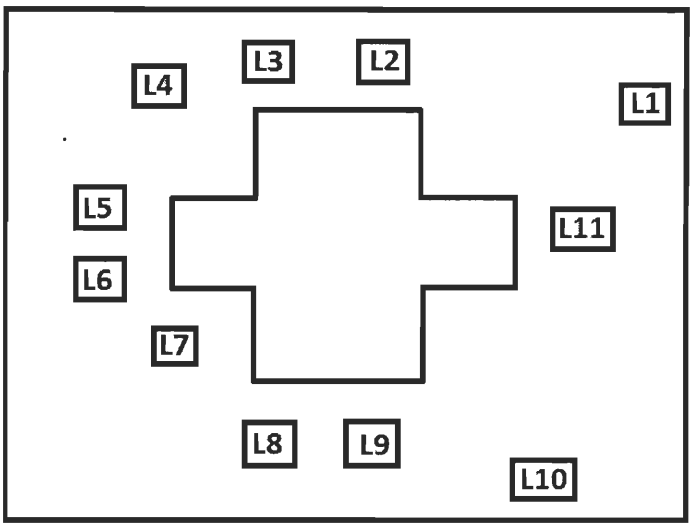


Figure 5-1– A hypothetical of construction site layout (Lien and Cheng 2012)

The frequency of trips matrix that expresses trips made in one day by project personnel is shown below.

Table 5-2 – Frequency matrix of trips between facilities for case study (Li and Love 2000)

Facility	A	B	C	D	E	F	G	H	I	J	K
A	0	5	2	2	1	1	4	1	2	9	1
B	5	0	2	5	1	2	7	8	2	3	8
C	2	2	0	7	4	4	9	4	5	6	5
D	2	5	7	0	8	7	8	1	8	5	1
E	1	1	4	8	0	3	4	1	3	3	6
F	1	2	4	7	3	0	5	8	4	7	5
G	4	7	9	8	4	5	0	7	6	3	2
H	1	8	4	1	1	8	7	0	9	4	8
I	2	2	5	8	3	4	6	9	0	5	3
J	9	3	6	5	3	7	3	4	5	0	5
K	1	8	5	1	6	5	2	8	3	5	0

The distance between possible locations on job site is expressed in matrix below. It is measured in meters between locations.

Table 5-3 - Travel distance matrix between facilities for case study (Li and Love 2000)

Location	1	2	3	4	5	6	7	8	9	10	11
1	0	15	25	33	40	42	47	55	35	30	20
2	15	0	10	18	25	27	32	42	50	45	35
3	25	10	0	8	15	17	22	32	52	55	45
4	33	18	8	0	7	9	14	24	44	49	53
5	40	25	15	7	0	2	7	17	37	42	52
6	42	27	17	9	2	0	5	15	35	40	50
7	47	32	22	14	7	5	0	10	30	35	40
8	55	42	32	24	17	15	10	0	20	25	35
9	35	50	52	44	37	35	30	20	0	5	10
10	30	45	55	49	42	40	35	25	5	0	10
11	20	35	45	53	52	50	40	35	10	10	0

5.2.2. Objective Function

Since Lien and Cheng (2012) and Li and Love (2000) only considered travel distance and frequency of trips between facilities, the same objective function as (5), (6), (7) and (8) is used for valid comparison.

5.2.3. Performance Comparison with PBA (Li and Love 2000)

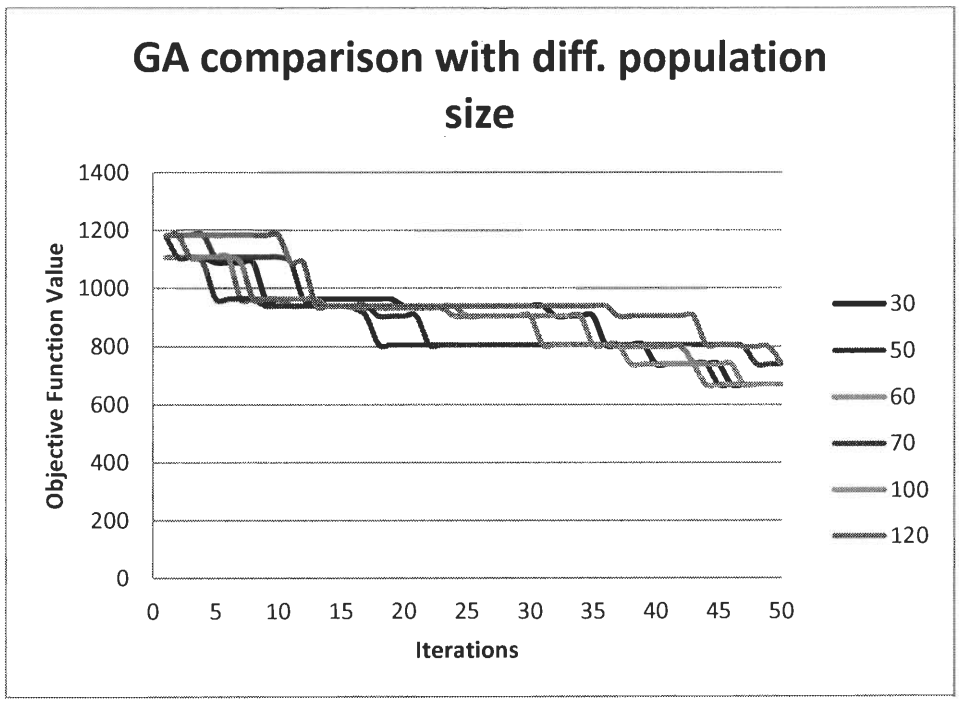


Figure 5-2 – Performance of genetic algorithm with various population sizes

Table 5-4 - Data of performance of genetic algorithm with various population sizes

	Population					
	30	**50**	**60**	**70**	**100**	**120**
StdDev	139	159.5	123.7	100.7	136.5	123.9
StdErr	19.7	22.6	17.5	14.2	19.3	17.5
Mean	905	867.9	886.4	871.3	889.2	968.3
Worst	1182	1182	1106	1182	1182	1182

Table 5-5– Genetic algorithm parameters

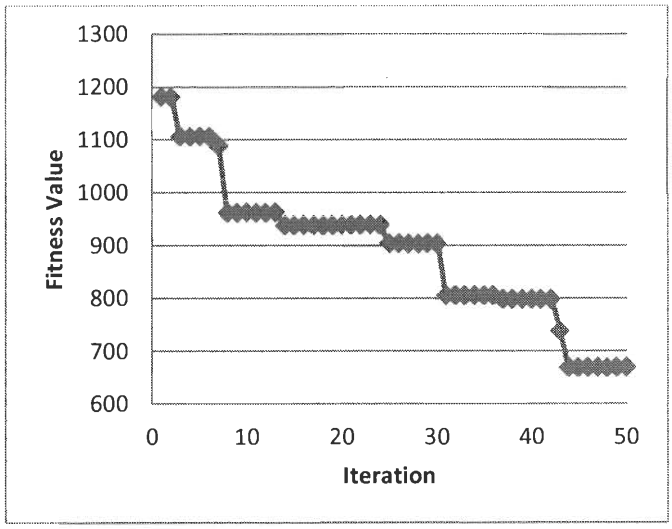
Maximum number of iterations	50
Population size	100
Probability of Selection	100%
Probability of Mutation	0.1%

5.2.4. Produced Layout Solution

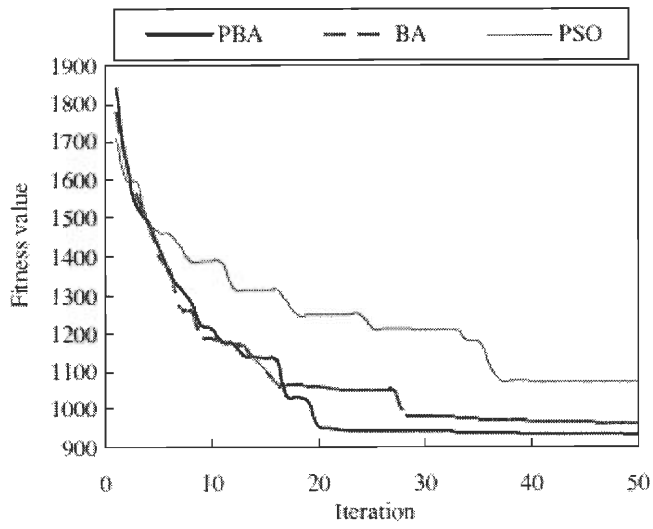
Table 5-6– String layout representation

Facilities	I	G	K	C	A	F	B	J	D	E	H
Locations	1	2	3	4	5	6	7	8	9	10	11

5.2.5. Performance Comparison with PBA (Lien and Cheng 2012)



(a)



(b)

Figure 5-3– (a) Performance of GA, (b) Performance of PBA, BA and PSO (Lien and Cheng 2012)

Table 5-7– Genetic algorithm vs. PSO, BA and PBA

	Mean	StdErr	Best	Worst
PBA	932.01	11.55	920.12	970.47
GA	889.2	19.3	669	1182
BA	963.92	30.04	920.12	1033
PSO	1072	316.7	928.47	2000

5.2.6. Results and Discussion

For maximum speed of evolution of chromosomes, all GA operators' parameters, except for the population size, were set at their maximum (Table 5-5). According to Li and Love (2000), the test of various population sizes was carried out in order to test the performance of GA. Based on Figure 5-2 and Table 5-4 the performance of the algorithm was best at mid-population of 100 with standard deviation of 136.5 and mean of 889.2. Therefore, the population size parameter was adjusted to 100 instead of initial 20. Figure 5-2 presents all data of GA performance based on different population sizes.

With data from Table 5-4, it is critical to look for the lowest standard error, highest standard deviation, lowest mean, and lowest worst cost. It is noticeable that at population size of 50, the result has standard error of 22.6, which is the highest in the table. That means it would require more iteration to produce the lowest cost of $669. With population of 70, it has the lowest standard deviation, which indicates that there were many overlapping results during the process. At population size of 60, it has the lowest worst-cost with a low standard deviation and standard error. Population size of 120 also has low standard deviation. Therefore, 4 parameters of population sizes 50, 60, 70 and 120 are ruled out. Two valuable population sizes are 30 and 100 because they have high standard deviation, reasonable standard error and mean. By observing these

two sets of data, it was noticed that the population size of 100 produced a lower mean, so it makes more sense to select 100 as the GA population size. The parameter of GA in Table 5-5 such as iterations, population size, and crossover, mutation rate were decided by data of case study 1 and to match with Li and Love (2000) and Lien and Cheng (2012) for validation.

From the above case study, the best solution was produced at $669 (from Table 5-7) which is lower than the hybrid swarm approach from Lien and Cheng (2012). This solution was generated within 9 min of CPU run time. Contrary to Li and Love (2000) and Lien and Cheng (2012), this paper considered replacing the side and main gates with 2 different facilities since the gates were fixed and not taken into account in the algorithm. Since the best cost was reached at a slower rate than others, the approach also produced the acceptable mean of $889.2. It is also important to observe the worst cost value: in this case, the total cost is $1182 compared to $970.5, $1033 and $2000 from PBA, BA and PSO, respectively.

An important performance measure of the algorithm is the standard error, which represents the accuracy of the sample mean as an estimate of the population mean as illustrated in Table 5-4. The standard error of GA with population size of 100 is 19.3, which means that the accuracy of 50 iterations with best solution is ± 19.3 to the real value with unknown amount of iterations. Therefore, the best total cost produced by the GA is $669 ± 19.3 compared to the estimate real value. On the other hand, the greater the standard deviation, the more the algorithm evolves the fitness value; thus, reducing the amount of overlapping solutions. The standard deviation of the GA was 136.5 which is the second highest (139 is the highest, which corresponds to population number of 30).

With standard error of 19.3 compared to 11.55 of PBA, it indicates that if both algorithms were run in infinite amount of time, GA would obtain the optimal solution slower than PBA.

This case study was selected from Lien and Cheng (2012) and Li and Love (2000) in order to validate performance GA for this study. It is clear that with different operators, GA performs equally or better than previous GA (Li and Love 2000), PBA, BA and PSO. With this case study, the objective function produces the minimum cost of $669 while minimizing travel distances between locations and frequency of trips between facilities made by personnel. These two requirements are very critical in site layout planning.

5.3. Case-Studies

Two case studies are developed to illustrate the functionality of the developed prototype systems. These case studies include two Western Michigan University campus projects namely: the new Sangren Hall and the Western Apartment View Projects. The case studies are described as follows:

5.3.1. Case Study 1: Sangren Hall Project

5.3.1.1. Overview

The objective of this case study is to demonstrate the implementation of shifting cost and setup cost of each facility in the objective function. As demonstrated above, the

perfomance of GA was validated by comparing the results with hybrid swarm algorithm (Lien and Cheng 2012). By adding shifting cost and setup cost (after each phase and throughout project duration), the total cost would be increased. The new Sangren hall project on Western Michigan University campus was used for the case study. The project involved an extension of the old Sangren hall on the main campus. Figure 5-5 shows the plan view of the job site. The existing building (old Sangren hall), which is at the north of construction site, would still be occupied during the school year. The green line in the figure shows the boundary of the site.

5.3.1.2. Number of Available Locations

There are a total of 12 available locations where temporary facilities can be located. These locations are shown on the plan view in Figure 5-5. The rectangular distance between these locations is calculated and presented in the travel distance matrix in Table 5-8. Note that these locations can always be changed by user reference. As the locations change, the travel distances must also be changed.

Table 5-8 – Travel distances between available locations

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	6	8	10	8	11	13	15	17	21	17
2	4	0	2	4	6	9	9	11	16	18	20	17
3	6	2	0	2	4	10	8	9	11	14	15	11
4	8	4	2	0	2	10	7	7	10	12	13	10
5	10	6	4	2	0	11	10	7	8	10	11	9
6	8	9	10	10	11	0	3	6	9	11	12	13
7	11	9	8	7	10	3	0	3	6	7	9	10
8	13	11	9	7	7	6	3	0	3	4	6	7
9	15	16	11	10	8	9	6	3	0	2	3	4
10	17	18	14	12	10	11	7	4	2	0	3	5
11	21	20	15	13	11	12	9	6	3	3	0	3
12	17	17	11	10	9	13	10	7	4	5	3	0

5.3.1.3. Frequency of Trips between Facilities

Since there are 12 available locations, the number of temporary facilities is 12 or less. For the purpose of this case study, there will be 12 facilities involved in each phase throughout the project life. The project schedule shown in Figure 5-4 is broken down into three critical phases: substructure, superstructure and interior finish as shown in Table 5-9. The highlighted facilities in phase 2 and 3 are new to the phases.

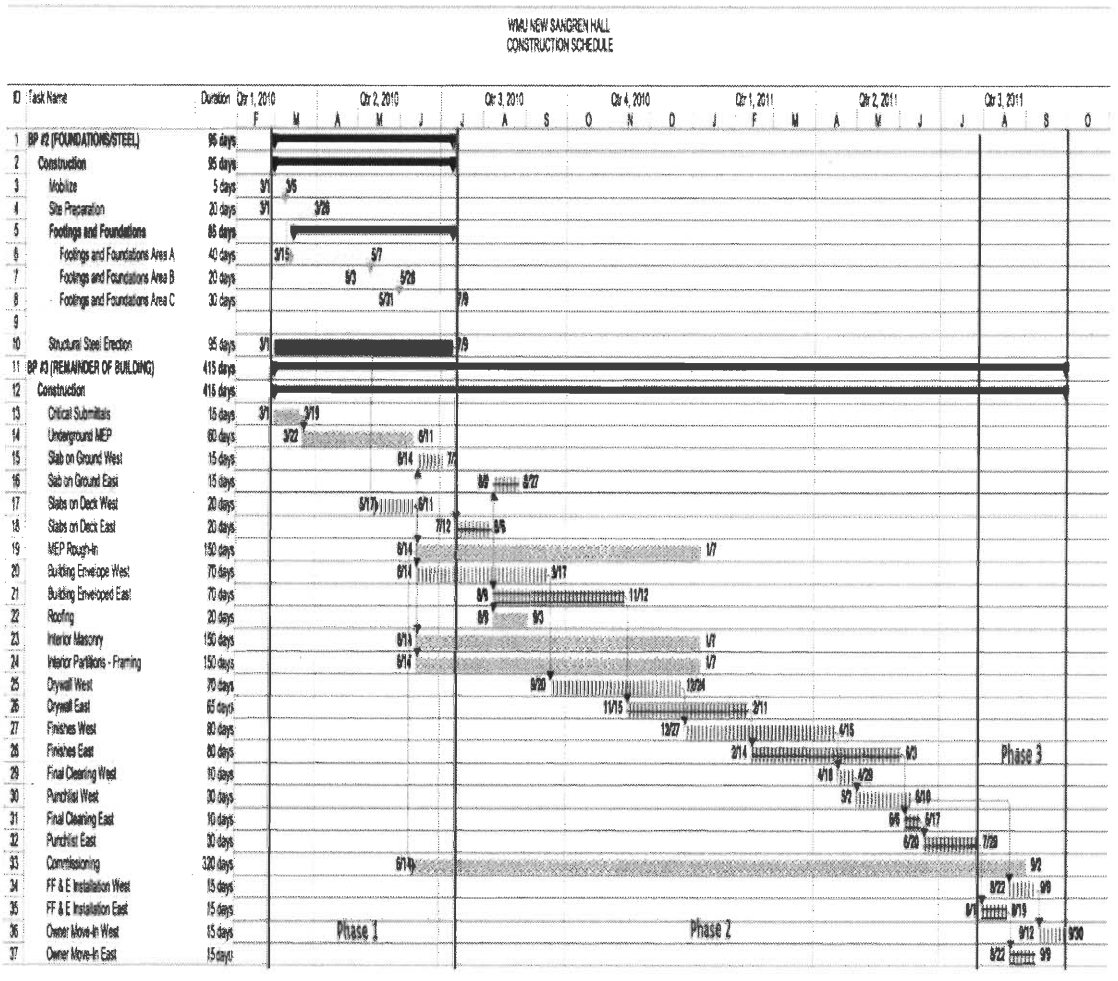


Figure 5-4 – Construction schedule

Table 5-9 – List of facilities involved throughout project lifetime

	Phase 1	Phase 2	Phase 3
	March 1st. 2010 to July 9th, 2010	July 9th, 2010 to August 6th, 2011	August 6th, 2011 to September 9th, 2011
A	Tool storage container	Tool storage container	Tool storage container
B	Steel bar and Half-finished products storage area	Steel bar and Half-finished products storage area	Steel bar and Half-finished products storage area
C	Molding Board storage are	Molding Board storage are	Molding Board storage are
D	Steel structure elements storage area	Steel structure elements storage area	Steel structure elements storage area
E	Concrete elements storage area	Concrete elements storage area	Material Hoist #2
F	Storage Container	Storage Container	Storage Container
G	Rebar bending yard	Rebar bending yard	Interior Material Storage
H	Mechanical and electrical equipment	Mechanical and electrical equipment	Mechanical and electrical equipment
I	Carpentry Workshop	Material Hoist #1	Material Hoist #1
J	Construction waste dumpster	Construction waste dumpster	Construction waste dumpster
K	Round Bottom Tank	Round Bottom Tank	Round Bottom Tank
L	Main contractor site office	Main contractor site office	Main contractor site office

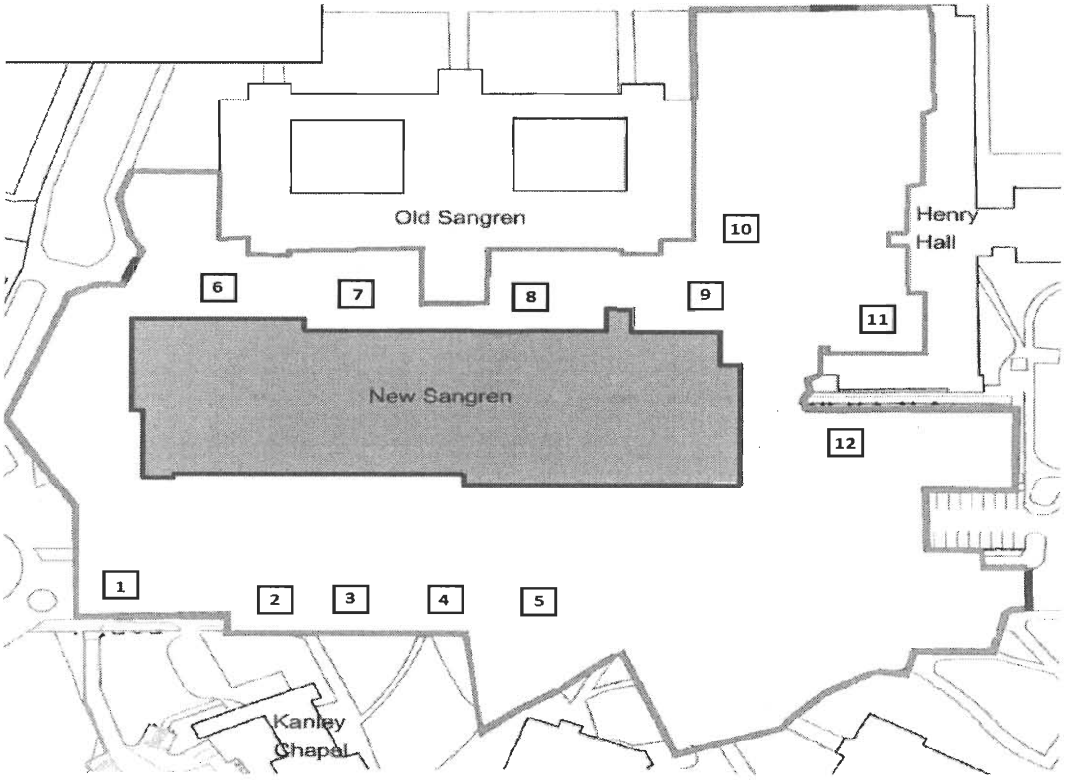


Figure 5-5– Plan view of job site and available locations

According to the construction schedule, there are different tasks to be executed and each task may require different or similar facilities within each project phase. Each phase will have different interaction, especially when new facilities come into place. Frequency matrix in Table 5-10 is measured in trips per day made by personnel. The ultimate goal is to minimize the total cost for site layout planning; therefore, any facilities that are the same between phases will be considered as fixed.

Table 5-10 – Frequency of trips between facilities of phase 1

	A	B	C	D	E	F	G	H	I	J	K	L
A	0	15	23	30	40	41	45	50	35	2	20	20
B	15	0	10	18	25	27	32	42	25	45	5	15
C	23	10	0	8	15	17	22	32	29	17	45	10
D	30	18	8	0	7	9	14	24	30	13	15	18
E	40	25	15	7	0	2	7	17	21	19	52	12
F	41	27	17	9	2	0	5	15	19	12	10	19
G	45	32	22	14	7	5	0	10	12	14	20	11
H	41	42	32	24	17	15	10	0	23	5	35	10
I	35	50	52	44	37	35	30	20	0	5	15	15
J	2	45	17	13	19	12	14	5	17	0	10	3
K	20	5	45	15	52	10	20	35	5	10	0	4
L	20	15	10	18	12	19	11	10	6	3	4	0

5.3.1.4. Setup Cost of Temporary Facilities

Setup cost of the temporary facilities may vary because the contracting company usually outsources such services. Table 5-11 shows an average set up cost for the facilities. It is often more economical renting the facilities than owning them. Therefore, the setup cost table was prepared based on interactions with the contracting company. This cost will be treated as being constant every time the site layout is generated. There is no guarantee that all facilities will be placed on site; therefore, only facilities involved would carry the setup cost.

Table 5-11 – Setup cost of temporary facilities

	Setup cost (US dollar)
Tool storage container	150
Steel bar and Half-finished products storage area	200
Molding Board storage are	200
Steel structure elements storage area	200
Concrete elements storage area	200
Storage Container	147
Rebar bending yard	150
Mechanical and electrical equipment	150
Carpentry Workshop	200
Construction waste dumpster	50
Round Bottom Tank	160
Main contractor site office	148
Material Hoist #1	100
Material Hoist #2	100
Interior Material Storage	150

5.3.1.5. Shifting Cost of Temporary Facilities

Shifting cost of temporary facilities is usually difficult to estimate. To eliminate this cost, construction managers usually try to limit the movement of the facilities. For the purpose of this case study, it is assumed that some facilities will have to be moved around the job site while keeping the total cost minimum. According to the contracting company, it is appropriate to assume the shifting labor cost of 22 US dollars per hour. Since the shifting cost cannot be calculated separately, it is assumed that it takes one hour to move each of these facilities to new locations based on given number of labor force. Table 5-12 presents an average shifting cost of the facilities based on the amount of labor required.

Table 5-12 – Shifting cost of temporary facilities

	Labor (person)	**Total labor cost (US dollar)**
Tool storage container	2	44
Steel bar and Half-finished products storage area	4	88
Molding Board storage are	4	88
Steel structure elements storage area	4	88
Concrete elements storage area	4	88
Storage Container	2	44
Rebar bending yard	3	66
Mechanical and electrical equipment	5	110
Carpentry Workshop	4	88
Construction waste dumpster	2	44
Round Bottom Tank	3	66
Main contractor site office	2	44
Material Hoist #1	3	66
Material Hoist #2	3	66
Interior Material Storage	3	66

5.3.1.6. Objective Function

Objective function is similar to equation (3.1), (3.2), (3.3) and (3.4) in section 3.1.1.

5.3.1.7. Produced Layout Solution

Table 5-13 – Phase 1 temporary facilities string layout presentation

Facilities	K	J	D	C	G	F	B	H	A	E	L	I
Locations	1	2	3	4	5	6	7	8	9	10	11	12

Table 5-14 – Total cost of each site layout generation

	Layout Cost	Setup Cost	Shifting Cost	Total Cost
Phase 1	$719	$1,955	---	$2,674
Phase 2	---	$100	$88	$188
Phase 3	---	$250	$154	$404

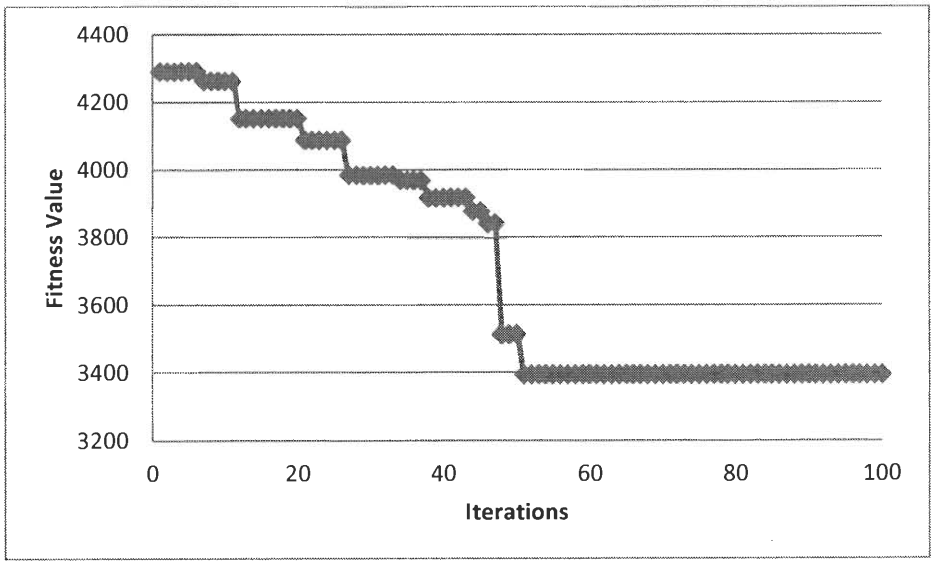


Figure 5-6 – GA performance for Sangren Hall project

According to Figure 5-6, total cost converged at about 50 iterations as validation stated. Once GA reaches its optimal solution, the data tends to remain nearly linear until it achieves its optimal solution.

5.3.1.8. Discussion

With all facilities needed to be setup at the beginning of the project, layout cost generated by GA was produced as $719 and setup cost was calculated as $1955. As mentioned above, there is no involvement of Material Hoist #1, #2 and Interior Material

Storage; therefore, their setup costs were left out. As the project continued to the 2nd phase, facility I (carpentry workshop) was replaced by Material Hoist #1. From Table 5-14, $100 was paid to setup Material Hoist #1 and shifting cost of $88 for moving the Carpentry Workshop away. The total cost for the 2nd phase of the project is $188. The project estimators considered the removal and shifting cost of the facilities as the same. In the 3rd phase of the project, facilities E and G, which are Concrete Element Storage Area and Rebar Bending Yard were replaced by Material Hoist #2 and Interior Material Storage, respectively. The setup cost of both facilities is $250 and shifting cost of the old facility is $154. At the initial stage of projects, site layout planning could be expensive since all facilities are required to be setup, but as the project goes on, it becomes less expensive.

5.3.2. Case Study 2: Western View Apartment Project

5.3.2.1. Overview

This case study demonstrated the automated site layout system on actual project, which is Western View Apartment on Western Michigan University campus. The total area of the project site is 136,800ft^2 (15,200 m^2). Figure 5-7 represents the plan view of proposed Western View Apartment project. The interested site layout area (red zone in Figure 5-7) is the proposed the parking lot which is north of building 306. This area has a size of 59,535ft^2 (6,615 m^2). Figure 5-8 shows the Navisworks model and associated schedule of this case study project. To enhance visibility and analysis, the interested zone

is scaled down to the laboratory area using a scale of 1:10. Therefore, the laboratory scale dimensions for the site boundary is 6.3 m by 10.5 m with a total area of 66.1 m^2.

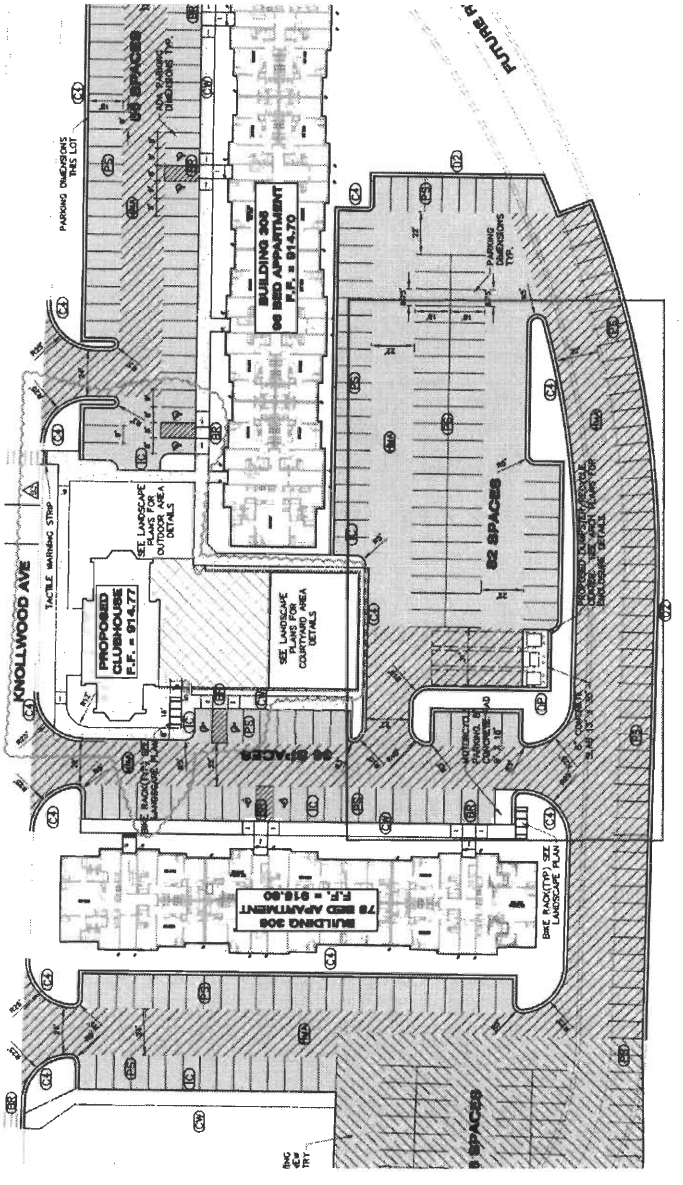


Figure 5-7 – Project site with proposed site layout area

Figure 5-8 – Building model with project schedule in Navisworks Manage

5.3.2.2. Number of Available Locations

The size of each possible location (where temporary facilities can be placed) is 1 m^2. This is equivalent to 10 m on the actual project site on a 1:10 scale. With an area of 66.1 m^2, there are a total 60 possible locations in this case study. A portion of interested area will tend to overlap with the proposed building (Building 306). This means that there will be 6 unavailable possible locations on the site; leaving a total of 54 possible locations (this serves an input to the RFID-RTLS system). Once this information has been entered into the RFID-RTLS system, the coordinates (X, Y) of each possible location is calculated and entered into the database. GA will then extract the information to create distances matrix between the locations. This matrix is required for computing the objective function. Figure 5-9 is the produce from RTLS system, which represents all possible locations for temporary facilities in the project.

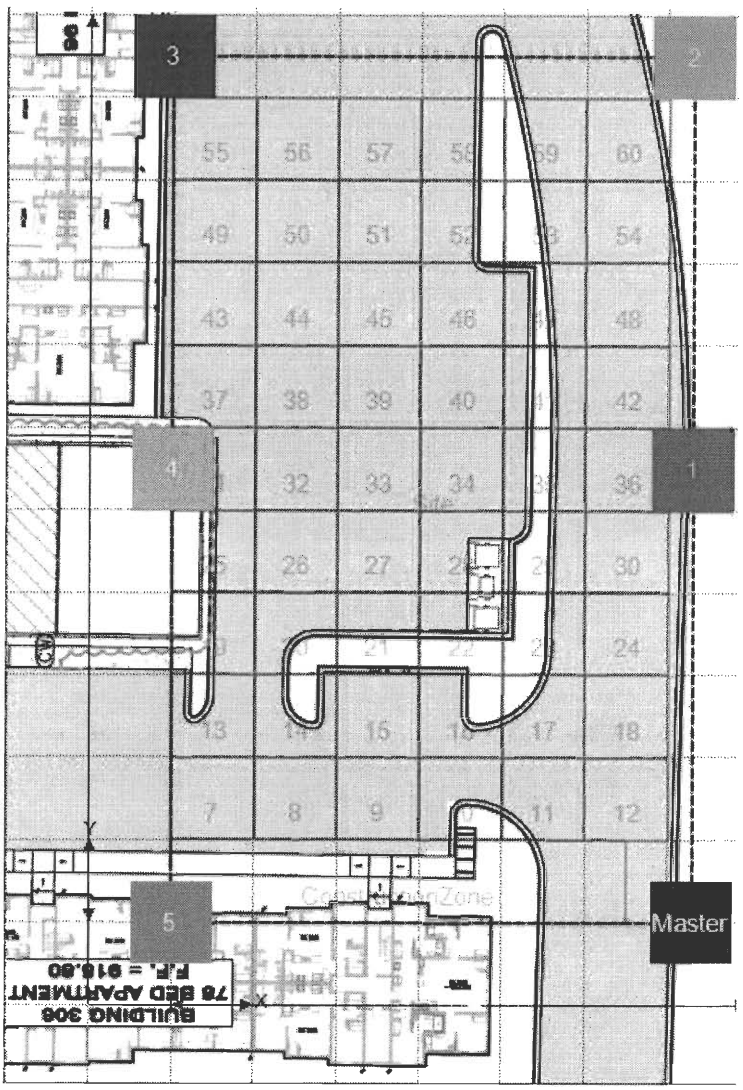


Figure 5-9 – Possible locations layout in RFID-RTLS system interface

5.3.2.3. Frequency of trips between facilities

Since there are 54 available locations, the number of temporary facilities in each phase of the project must be 54 or less. In this case study, there are 18 temporary facilities required throughout the project lifecycle. The project schedule shown in Figure 5-10 is broken down into three critical phases: substructure, superstructure and interior

finish. The project spans over 1 school year duration; for testing the system, the project was scaled down to 3 days.

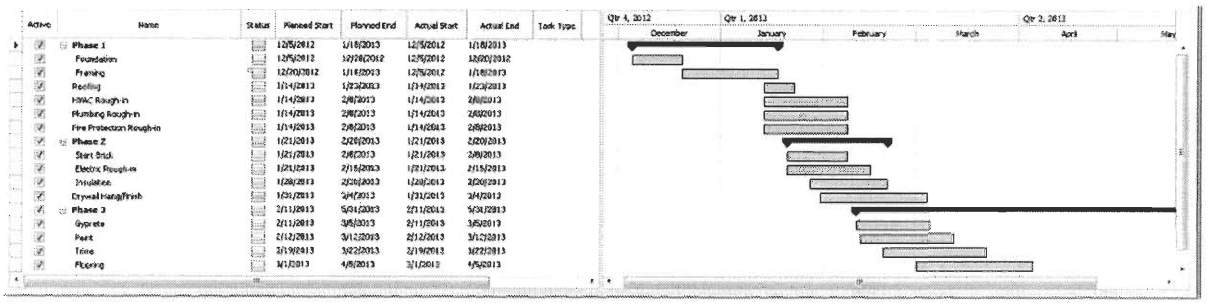


Figure 5-10 – Construction schedule

Table 5-15 - List of facilities involved throughout project lifetime

List of temporary facility	
ID	**Name**
1	Concrete Element storage area
2	Rebar bending yard
3	Carpentry workshop
4	Molding board storage area
5	Storage Container
6	Material delivery storage area
7	Mechanical equipment storage
8	Raw material container
9	Material hoist
10	Brick Yard
11	Electrical equipment storage area
12	Insulation material area
13	Drywall sheet storage
14	Concrete pour equipment
15	Paint storage
16	Finishing equipment storage
17	Carpet/interior material storage
18	Cleaning equipment

Table 5-16 – List of temporary facilities involving in each construction phase

Temporary Facilities	Phase 1	Phase 2	Phase 3
	1	3	13
	2	4	14
	3	5	15
	4	6	16
	5	7	17
	6	8	18
	7	9	
	8	10	
		11	
		12	
		13	

According to the project schedule, there are different tasks to be executed and each task may require one or more facilities within each project phase. Table 5-15 illustrates the list of temporary facilities involved in each phase of the project. The ultimate goal is to minimize the total cost for site layout; therefore, any facilities that are the same between phases will be considered as fixed. Each phase will have different interaction between facilities, especially when new facilities are considered. Table 5-16 represents certain set of temporary facilities involved in each phase. The frequency of trips matrix in Table 5-17 is assumed and measured in trips per day made by personnel. Although Table 5-17 contains the interaction of all 18 temporary facilities, GA only calls out the frequency values of facilities within the current phase. For example, phase 1 has facilities 1 to 8; therefore, the frequency of trips matrix for the phase will be 8-by-8 matrix.

Table 5-17 – Frequency of trips between facilities

	Frequency of trips between temporary facilities																	
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
1	0	7	23	16	7	7	6	11	21	15	10	11	9	12	12	7	11	5
2	7	0	25	21	24	23	17	10	17	25	8	16	9	17	6	24	5	5
3	23	25	0	14	8	22	10	7	13	9	9	14	24	10	24	9	6	22
4	16	21	14	0	9	14	10	8	13	10	17	20	21	5	19	24	11	21
5	7	24	8	9	0	7	10	20	9	11	22	24	5	16	22	17	14	21
6	7	23	22	14	7	0	20	17	9	14	8	8	10	9	15	9	13	21
7	6	17	10	10	10	20	0	7	18	19	15	8	10	19	9	20	24	9
8	11	10	7	8	20	17	7	0	8	6	7	9	17	10	20	8	13	6
9	21	17	13	13	9	9	18	8	0	14	12	20	9	20	15	12	14	18
10	15	25	9	10	11	14	19	6	14	0	23	10	22	9	23	21	25	19
11	10	8	9	17	22	8	15	7	12	23	0	21	11	11	12	11	22	8
12	11	16	14	20	24	8	8	9	20	10	21	0	7	18	12	12	15	21
13	9	9	24	21	5	10	10	17	9	22	11	7	0	9	18	8	17	18
14	12	17	10	5	16	9	19	10	20	9	11	18	9	0	7	18	22	8
15	12	6	24	19	22	15	9	20	15	23	12	12	18	7	0	10	6	14
16	7	24	9	24	17	9	20	8	12	21	11	12	8	18	10	0	15	11
17	11	5	6	11	14	13	24	13	14	25	22	15	17	22	6	15	0	20
18	5	5	22	21	21	21	9	6	18	19	8	21	18	8	14	11	20	0

5.3.2.4. Setup Cost of Temporary Facilities

Setup cost as explained in section 5.2.1.4 was prepared based on interactions with the contracting company and only facilities involved would carry the setup cost. Table 5-18 represents an average cost for mid-size projects.

Table 5-18 – Setup cost of temporary facilities

ID	Name	Setup Cost ($)
1	Concrete Element storage area	200
2	Rebar bending yard	150
3	Carpentry workshop	200
4	Molding board storage area	200
5	Storage Container	130
6	Material delivery storage area	110
7	Mechanical equipment storage	125
8	Raw material container	120
9	Material hoist	100
10	Brick Yard	160
11	Electrical equipment storage area	145
12	Insulation material area	135
13	Drywall sheet storage	160
14	Concrete pour equipment	165
15	Paint storage	170
16	Finishing equipment storage	120
17	Carpet/interior material storage	130
18	Cleaning equipment	100

5.3.2.5. Shifting Cost of Temporary Facilities

Section 5.2.1.5 detailed how the shifting cost was calculated. The same process is used in this case study for all temporary facilities. Table 5-19 presents an average shifting cost of the facilities based on the amount of labor required. This data was collected through detailed interaction with the contracting company.

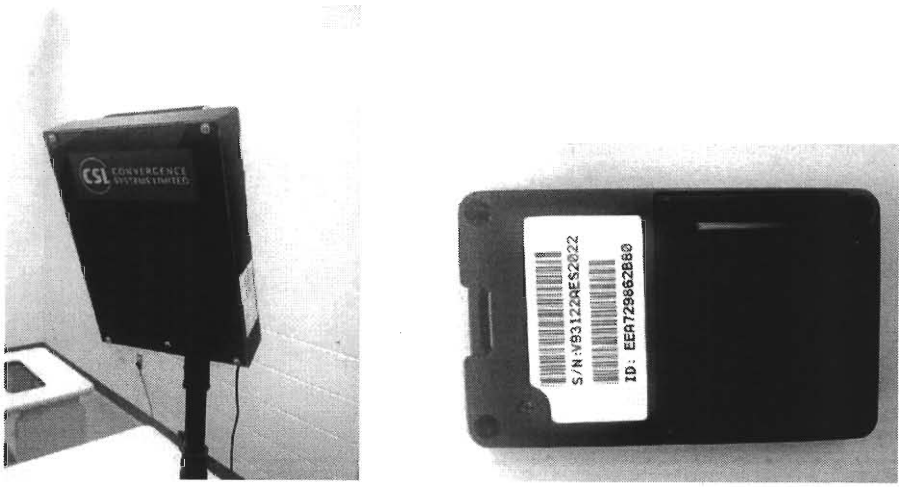
Table 5-19 – Shifting cost of temporary facilities

ID	Name	Labor (person)	Cost ($)
1	Concrete Element storage area	4	88
2	Rebar bending yard	3	66
3	Carpentry workshop	4	88
4	Molding board storage area	4	88
5	Storage Container	4	88
6	Material delivery storage area	3	66
7	Mechanical equipment storage	3	66
8	Raw material container	5	110
9	Material hoist	2	44
10	Brick Yard	6	132
11	Electrical equipment storage area	3	66
12	Insulation material area	4	88
13	Drywall sheet storage	3	66
14	Concrete pour equipment	5	110
15	Paint storage	3	66
16	Finishing equipment storage	5	110
17	Carpet/interior material storage	4	88
18	Cleaning equipment	2	44

5.3.2.6. Objective Function

Objective function is similar to equation (3.1), (3.2), (3.3) and (3.4) in section 3.1.1.

5.3.2.7. Laboratory Sample Setup and Equipment



(a) (b)

Figure 5-11 – (a) Master/Slave readers; (b) RFID tag

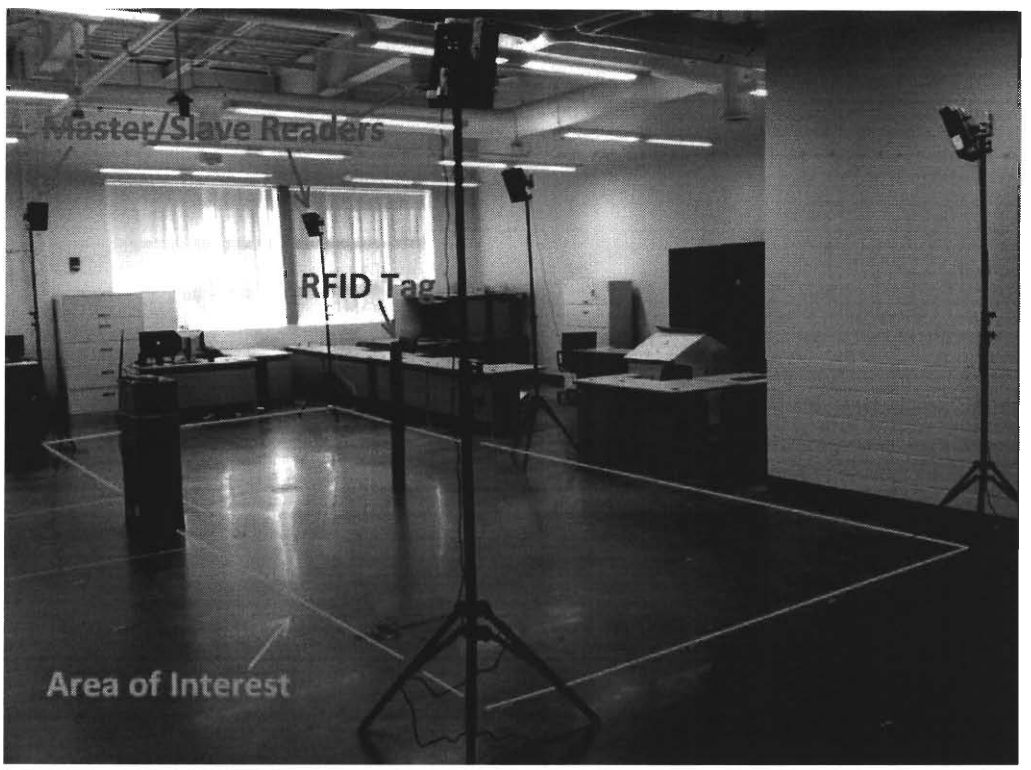
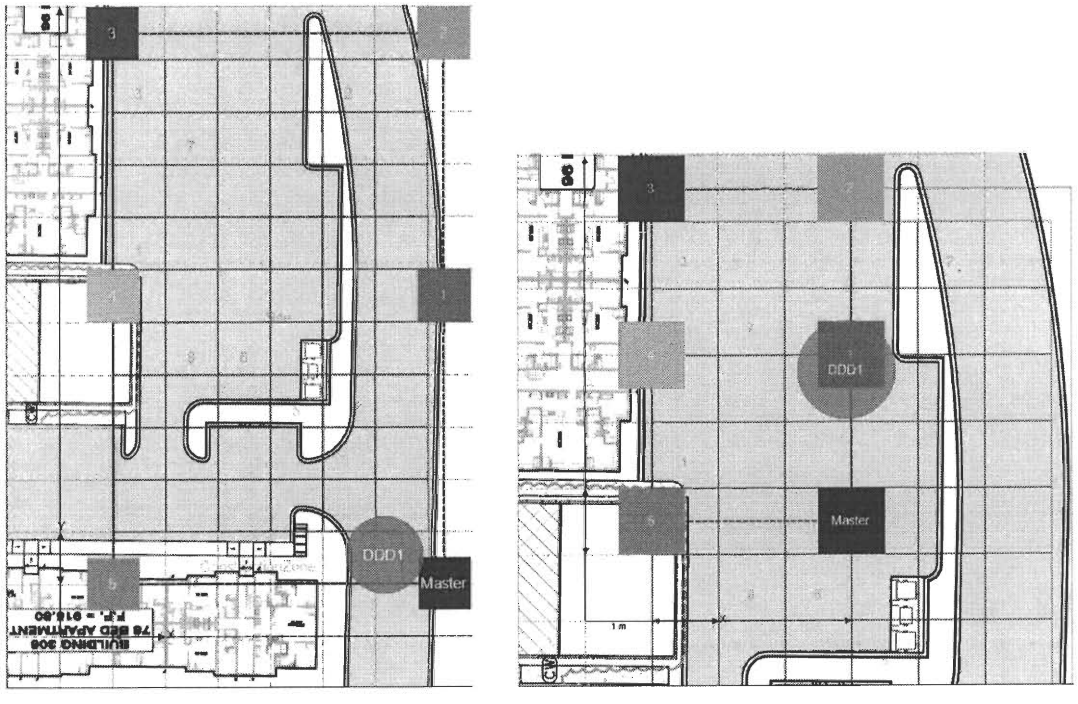


Figure 5-12 – Sample RTLS System setup in laboratory

5.3.2.8. Results

Table 5-20 – Phase 1 string layout representations

Facility	5	4	8	6	1	7	3	2
Locations	22	25	26	27	37	50	55	59

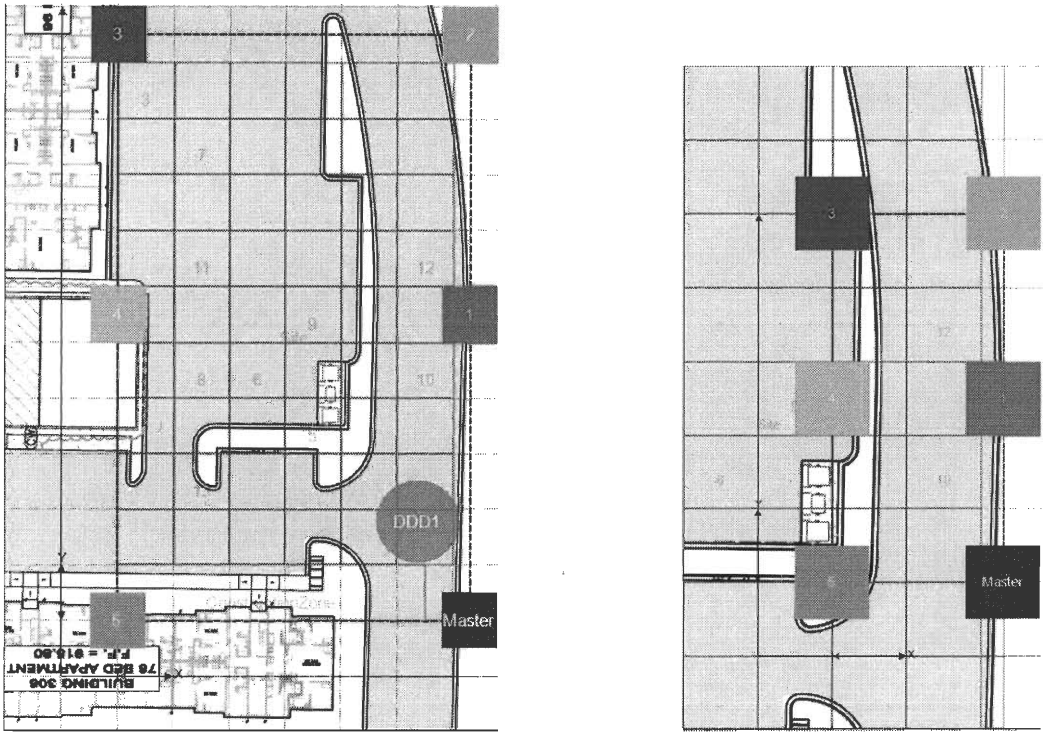


(a) (b)

Figure 5-13 – (a) Site layout of Phase 1 in RFID-RTLS system interface (b) Zoom-in interested facilities on site.

Table 5-21 – Phase 2 string layout representations

Facilities	13	5	4	8	6	10	9	11	12	7	3
Locations	14	22	25	26	27	30	34	38	42	50	55



(a) (b)

Figure 5-14 - (a) Site layout of Phase 2 in RFID-RTLS system interface (b) Zoom-in interested facilities on site

Table 5-22 - Phase 3 string layout representations

Facilities	17	18	13	15	14	16
Locations	7	10	14	22	42	50

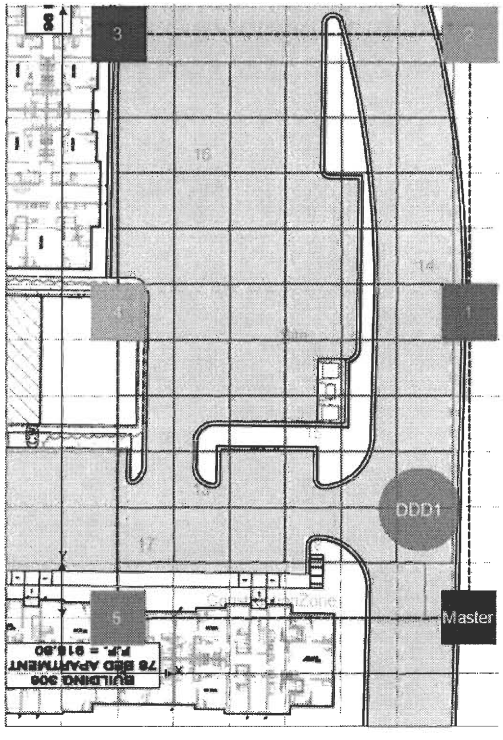


Figure 5-15 – Site layout of Phase 3 in RFID-RTLS system interface

Table 5-23 – Total cost of each site layout generation

Phase	Layout Cost ($)	Setup cost ($)	Shifting Cost ($)	Total cost ($)
1	157.9	1235	---	1392.9
2	128.4	700	154	982.4
3	155.9	685	836	1676.9

5.3.2.9. Summary and Discussion

The test was carried out within a 3-day timeframe. According to the project schedule, at the beginning of phase 1, a total of 8 facilities were setup. GA generated a site layout with total cost of $1392.9. This total cost includes layout and setup cost of $157.9 and $1235, respectively. There was no shifting cost of temporary facilities in phase 1 since the facilities were only set up. The layout of the temporary facilities was produced and input into RFID-RTLS system. Figure 5-13 (a) shows the first layout on

RFID program interface. Construction personnel can then easily setup facilities based on Figure 5-13 (a).

After site layout generation, the RFID-RTLS system monitors the movement of tagged material(s) within the layout spaces on site. One tag (DDD1) was used for this case study. As in Figure 5-13, tag DDD1 is not within any of facilities, so the system will understand that all facilities are empty. Phase 1 has duration of 1 day, so it will not trigger GA for the next site layout generation until the end of the duration.

According to the project schedule, the last 4 tasks in phase 1 extend to phase 2; therefore, any facility associated with these tasks were considered fixed. When the duration of phase 1 elapsed, BIM triggered GA for new site layout. By taking the fixed facilities into account, GA only generated site layout based on the new facilities (9 to 13) in phase 2. The total cost of site layout in phase 2 is $982.4. This includes layout cost, setup cost and shifting cost: $128.4, $700 and $154, respectively. Setup cost represents the cost of setting up facilities 9 to 13 since they are new in phase 2. Shifting cost consists of facility 1 and 2 because they are no longer in use in phase 2. In addition, assuming the facilities are to be moved, the removal cost will be added to the shifting cost.

The same procedure as phase 1 and 2 took place in phase 3. According to the project schedule, facility 13 will not be moved since the associated task started in phase 2 and carried over to phase 3. Duration of phase 2 was also 1 day for testing purposes. BIM triggered GA for site layout generation after 1 day and the result is shown in Figure 5-14 and Table 5-21.

The total cost of site layout generation in phase 3 is $1676.9. This includes the layout cost, setup cost and shifting cost: $155.9, $685 and $836, respectively. The setup cost includes the cost of setting up facilities 14 to 18 and the shifting cost consists of the cost of shifting or moving facilities 3 to 12. The total cost heavily depends on the number of facilities associated with each phase and the number of facilities that are shifted. From a practical point of view, while most of facilities are initially set up, it is important to avoid moving the facilities between phases so as to minimize the total layout cost. However, the objective of this case study is to demonstrate the capability of the automated site layout system generating site layout based on the project schedule and space availability. The system provides the optimal layout solution based on minimal total cost (material transportation cost, setup and shifting cost, etc.). The system also enables visualizing the generated layout through the RFID-RTLS system graphical user interface.

CHAPTER 6: SUMMARY AND CONCLUSIONS

6.1. Introduction

This chapter presents a general summary of the overall research, draws conclusions and makes recommendations for further research. This research focuses on the development of an automated adaptive system for construction site layout generation. The research resulted in the development of software and system architecture of the resulting prototype system. The research also presented the implementation of the developed prototype system on a sample campus project. The chapter presents a summary and conclusion on the development and implementation of the prototype. This chapter also includes recommendations for future research.

6.2. General Summary

The section reviews the aim and objectives of this research and compares it with the research undertaken. The aim of the research was to develop an adaptive automated component level system for construction site layout generation. The specific objectives of the research project were:

- To identify the requirements for effective site layout generation. This requirements will be developed through an extensive review of literature and informal interactions with project managers;

- To investigate the potential of genetic algorithm (GA) for generating the optimum layout based on the identified requirements. This will include formulating the site layout problem. The formulated problem will be implemented in GA and validated using a hybrid swarm approach;
- To develop a systems and software architecture for the adaptive automated component level site layout generation system;
- To develop and experiment with automated site layout generation on laboratory scale prototypes. This experiment will enable tracking the status of key components and automatic generation of site layouts using the laboratory scale physical prototype;
- To test and validate the developed prototype on a sample case study.

The specific tasks undertaken in this research, with respect to the research objectives, are summarized below:

- ***Objective 1: To identify the requirements for effective site layout generation. This requirements will be developed through an extensive review of literature and informal interactions with project managers:***

A review was undertaken of the existing computational approaches to site layout planning. There were many optimization methods, but Genetic Algorithm is most suitable for solving site layout planning problems because it can be easily modified and does not require determination of problem search space. Extensive review was also conducted to identify requirements for effective site layout planning. These requirements include: minimizing frequency of trips between temporary facilities, minimizing travel distance

between locations, minimizing material transportation cost and minimizing setup and shifting cost of facilities between locations. The enabling technologies for various integration approaches in both the construction industry and other industry sectors were also reviewed; this include component and image based tracking technologies. In future research, image based technologies can be implemented into the system to track bulk materials such as sand, gravels, etc. RFID-RTLS system was most suitable in this research as component based tracking technology because of its ability to map and track components within areas of interest.

- ***Objective 2: To investigate the potential of genetic algorithm (GA) for generating the optimum layout based on the identified requirements. This will include formulating the site layout problem. The formulated problem will be implemented in GA and validated using a hybrid swarm approach:***

Based on the requirements, selected factors affecting the site layout cost were identified. These factors were used to formulate the site layout problem. The formulated problem was also implemented in GA. Details of the problem formulation and GA implementation is described in Chapter 3.

- ***Objective 3: To develop a systems and software architecture for the adaptive automated component level site layout generation system;***

Based on the problem formulation in Objective 2 and the software and systems architecture was developed in Chapter 4 of this thesis. The system integrates RFID-RTLS system, BIM and GA.

- ***Objective 4: To develop and experiment with automated site layout generation on laboratory scale prototypes. This experiment will enable tracking the status of key components and automatic generation of site layouts using the laboratory scale physical prototype;***

Two systems were developed and experimented with in the laboratory using small scale laboratory prototypes. A second system for automated site layout generation using an RFID-RTLS system was also developed. The details of the development and implementation of the prototype systems are presented in Chapter 5.

- ***Objective 5: To test and validate the developed prototype on a sample case study.***

Prototype #1 was validated with a similar hybrid swarm system prior to implementation on a sample project. Prototype #2 was also tested using a sample project as a case study.

6.3. Conclusions

Based on the findings outlined in the preceding chapters of the thesis, the following conclusions can be drawn:

- Genetic Algorithm was selected as the optimization method to solve the site layout planning problem. Validation of Genetic Algorithm was carried out and it is proved to be a reliable tool to generate site layout;

- For an effective site layout planning, the following factors will need to be considered: frequency of trips, travel distance between locations, material transportation cost, setup and shifting cost of facilities;
- Prototype #1 was developed to validate the performance of GA with multiple population size. The result of prototype #1 was also compared with results from PBA, BA and PSO. The result from GA was significantly improved with slightly higher standard error.
- The review of various enabling technologies for integration approaches in various industry sectors indicates that the RFID-RTLS system provides a high level of support for automated site layout generation. This is because of its capability for both spatial mapping and real-time location tracking;
- The development of prototype #2 was to demonstrate the integration of RTLS system and BIM with prototype #1 to create the automatic site layout system. The complete system really understands and adapts to current situation on site and project schedule in order to produce site layout.
- The developed prototype systems demonstrated the potential of sensing technologies and other computing systems for automated site layout generation. These systems offer opportunities for enhancing construction progress tracking, situation awareness on the job site resource tracking and access to site layouts (or subsets thereof).

6.4. Research Limitations

There are a number of limitations to the research described in this thesis. These are highlighted below:

- In the problem formulation only a single objective was considered. Opportunity exist for considering other objectives such as safety;
- Only limited field/outdoor testing of the developed system was undertaken. There is a need to demonstrate the practical functionality of the systems on a real construction site before robust conclusions can be drawn on the suitability of the approach to full-scale construction projects. This would also help to establish the organizational and other project-specific constraints that will need to be addressed;
- The developed system is only applicable to tracking components using the RFID-RTLS system. There is need to extend this system to the use of image based tracking systems such as digital cameras and laser scanners.

6.5. Recommendations for Further Research and Development

The research has identified a number of areas for further research. The key recommendations for further research and development are as follows:

- Different types of genetic operators perform differently within GA based on types of problem. Other operators could be considered to determine their effect on the solution;

- Multi-objective function can be considered to address more requirements of site layout planning;
- Other methods such as fuzzy method can be implemented in GA for qualitative variables;
- Instead of generating the site layout per phase, future studies could investigate the use of appropriate algorithm to investigate the optimal time to generate layouts;
- Future research could investigate strategies for tracking other material types for automated site layout generation. This will involve investigating a more comprehensive system rather than the component-based approach demonstrated in this research.

6.6. Concluding Remarks

This research has shown how construction site layouts can be automatically generated based on integrating BIM and sensing technologies with the physical construction process. The developed systems illustrate how construction site layouts can automatically generate based on the project schedule, context of the components on site and availability of the site spaces. These benefits include access to real-time status information, which will aid the construction project team in quick decision making and potentials for enhancing real-time construction site layouts. Another benefit is that this provides an opportunity for the project manager to understand the cost implications of different layout options throughout the project duration.

REFERENCES

Abdinnour-Helm, S., and Hadley, S. W. (2000). "Tabu search based heuristics for multi-floor facility layout." *International Journal of Production Research*, 38(2), 365-383.

Abeid, J., Allouche, E., Arditi, D., and Hayman, M. (2003). "PHOTO-NET II: a computer-based monitoring system applied to project management." *Automation in construction*, 12(5), 603-616.

Akanmu, A. A., Anumba, C. J., Messner, J. I., Lee, S., and Kundur, D. (2012). "Experimental Investigation of the Integration of Virtual Models and Physical Construction." *7th International Conference on Innovation in Architecture*, Engineering and Construction (Brazil, August 2012).

Akinci, B., Boukamp, F., Gordon, C., Huber, D., Lyons, C., and Park, K. (2006). "A formalism for utilization of sensor systems and integrated project models for active construction quality control." *Automation in Construction*, 15(2), 124-138.

Andayesh, M., and Sadeghpour, F. (2011). "Dynamic site layout planning through minimization of total potential energy." *Automation in Construction*, 31, 92-102.

Azhar, S., Hein, M., and Sketo, B. "Building information modeling (BIM): Benefits, risks and challenges." *Proc., Proceedings of the 44th ASC National Conference*.

Baykasoglu, A., Ozbakir, L., and Tapkan, P. (2007). "Artificial bee colony algorithm and its application to generalized assignment problem." *Swarm Intelligence: Focus on Ant and particle swarm optimization*, 113-144.

Bosche, F., and Haas, C. (2008). "Automated retrieval of 3D CAD model objects in construction range images." *Automation in Construction*, 17(4), 499-512.

Brilakis, I., and Soibelman, L. "Comparison of manual and user guided methodologies for the classification and retrieval of construction site images." *Proc., Construction Research Congress*, 5-7.

Chandratre, K., and Nandurkar, K. (2011). "Applying Genetic Algorithm to Dynamic Layout Problem." *International Journal of Applied*, 1(3), 1-9.

Cheung, S.-O., Tong, T. K.-L., and Tam, C.-M. (2002). "Site pre-cast yard layout arrangement through genetic algorithms." *Automation in Construction*, 11(1), 35-46.

Dickinson, M., Bearman, G., Tille, S., Lansford, R., and Fraser, S. (2001). "Multi-spectral imaging and linear unmixing add a whole new dimension to laser scanning fluorescence microscopy." *Biotechniques*, 31(6), 1272-1279.

Eberhart, R. C., and Shi, Y. (2004). "Particle Swarm Optimization." *IEEE Trans. Evol. Comput*, 8(3), 201-203.

El-Gafy, M., Abdelhamid, T., and Ghanem, A. (2010). "Using Simulated Annealing For Layout Planning of Construction Sites."

El-Omari, S., and Moselhi, O. (2009). "Data acquisition from construction sites for tracking purposes." *Engineering, Construction and Architectural Management*, 16(5), 490-503.

El-Rayes, K., and Said, H. (2009). "Dynamic site layout planning using approximate dynamic programming." *Journal of Computing in Civil Engineering*, 23(2), 119-127.

Ergen, E., and Akinci, B. "An overview of approaches for utilizing RFID in construction industry." *Proc., RFID Eurasia, 2007 1st Annual*, IEEE, 1-5.

Fard, M. G., Sridharan, A., Lee, S., and Peña-Mora, F. "Visual representation of construction progress monitoring metrics on time-lapse photographs." *Proc., Proc. Construction Management and Economics Conference, Reading, UK.*

Goldberg, D. E. (1989). "Genetic algorithms in search, optimization, and machine learning."

Goodrum, P. M., McLaren, M. A., and Durfee, A. (2006). "The application of active radio frequency identification technology for tool tracking on construction job sites." *Automation in Construction*, 15(3), 292-302.

Gordon, S., and Lichti, D. (2004). "Terrestrial laser scanners with a narrow field of view: the effect on 3D resection solutions." *Survey Review*, 37(292), 448-468.

Hegazy, T., and Elbeltagi, E. (1999). "EvoSite: Evolution-based model for site layout planning." *Journal of computing in civil engineering*, 13(3), 198-206.

Hergunsel, M. F. (Sept 2011). "Benefits of Building Information Modeling for Construction Managers and BIM Based Scheduling." Worcester Polytechnic Institute.

Jaselskis, E. J., Cackler, E. T., Walters, R. C., Zhang, J., and Kaewmoracharoen, M. (2006). "Using scanning lasers for real-time pavement thickness measurement."

Kiziltas, S., Burcu, A., Ergen, E., and Pingbo, T. (2008). "Technological assessment and process implications of field data capture technologies for construction and facility/infrastructure management." ITcon.

Lam, K. C., Ning, X., and Ng, T. (2007). "The application of the ant colony optimization algorithm to the construction site layout planning problem." *Construction Management and Economics*, 25(4), 359-374.

Li, H., and Love, P. E. (1998). "Site-level facilities layout using genetic algorithms." *Journal of Computing in Civil Engineering*, 12(4), 227-231.

Li, H., and Love, P. E. (2000). "Genetic search for solving construction site-level unequal-area facility layout problems." *Automation in Construction*, 9(2), 217-226.

Lien, L.-C., and Cheng, M.-Y. (2012). "A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization." *Expert Systems with Applications*.

Maier, H. R., Simpson, A. R., Zecchin, A. C., Foong, W. K., Phang, K. Y., Seah, H. Y., and Tan, C. L. (2003). "Ant colony optimization for design of water distribution systems." *Journal of water resources planning and management*, 129(3), 200-209.

Mawdesley, M. J., Al-Jibouri, S. H., and Yang, H. (2002). "Genetic algorithms for construction site layout in project planning." *Journal of construction engineering and management*, 128(5), 418-426.

Ning, X., Lam, K.-C., and Lam, M. C.-K. (2010). "Dynamic construction site layout planning using max-min ant system." *Automation in Construction*, 19(1), 55-65.

Oglesby, C. H., Parker, H. W., and Howell, G. A. (1989). *Productivity improvement in construction*, McGraw-Hill New York.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., and Zaidi, M. "The bees algorithm–a novel tool for complex optimisation problems." *Proc., Proceedings of IPROMS 2006 conference*, 454-461.

Sadeghpour, F., Moselhi, O., and Alkass, S. "Dynamic planning for site layout." *Proc., Proc., 30th Annual Conf. of Canadian Society of Civil Engineering*, CSCE.

Selvi, V., and Umarani, R. (2010). "Comparative analysis of ant colony and particle swarm optimization techniques." *International Journal of Computer Applications*, 5(4).

Song, B., and Mitchell, C. J. "RFID authentication protocol for low-cost tags." *Proc., Proceedings of the first ACM conference on Wireless network security*, ACM, 140-147.

Su, Y., Hashash, Y., and Liu, L. (2006). "Integration of construction as-built data via laser scanning with geotechnical monitoring of urban excavation." *Journal of Construction Engineering and Management*, 132(12), 1234-1241.

Teizer, J., Kim, C., Haas, C. T., Liapi, K. A., and Caldas, C. H. (2005). "Framework for real-time three-dimensional modeling of infrastructure." *Transportation Research Record: Journal of the Transportation Research Board*, 1913(1), 177-186.

Yagi, J., Arai, E., and Arai, T. (2005). "Parts and packets unification radio frequency identification (RFID) application for construction." *Automation in Construction*, 14(4), 477-490.

APPENDIX

MATLAB Code

Main file:

```
a = timer('ExecutionMode', 'fixedRate','TimerFcn', 'func(1)', 'Period',
60);
start(a);
```

Function TimerFcn:

```
%% Timer checking with Database Condition. Note that 1 is true, 0 is
false
function [x] = func(y)
clc

%% Connecting to MySQL Dtatabase
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Checking execution status from RFID-RTLS system and BIM
db.prepareStatement('SELECT Value FROM TriggerTable WHERE ID=1');
x = db.query();
x = cell2mat(struct2cell(x));
% db.prepareStatement('SELECT Value FROM TriggerTable WHERE ID=2');
% z = db.query();
% z = cell2mat(struct2cell(z));

%% Loop starting for phase updating, resetting status and triggering
SLP MATLAB
if x == y
    % Updating phase
    db.prepareStatement('SELECT Value FROM TriggerTable WHERE ID=4');
    j = db.query();
    j = cell2mat(struct2cell(j));
    j = j + 1;
    db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE
ID = "{Si}"', j, 4);
    db.query();
    db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE
ID = "{Si}"', 0, 1);
    db.query();

    % Triggering SLP MATLAB
    if j == 1
        slp1
    elseif j == 2
        slp2
    elseif j == 3
```

```
        slp3
    end
else
    disp('same')
end
db.close();
```

File slp1:

```
clear
clc
global iga x y
%% Connecting to MySQL and extract coordinates of available locations
%% generated from RTLS and C# code
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
db.close()
z = cell2mat(struct2cell(z));
mainfren = ones(z);
mainfren(logical(eye(size(mainfren)))) = 0;
m = z;

%% Setup the GA Parameters
ff='slpfun1';                          %Objective function
fac=m;                                 %Number of facilities
loc=m;                                 %Number of locations
maxit=50;                              %Maximum number of iterations
popsize=100;                           %Population Size
mutation=0.01;                         %Set mutation rate
selection=1;                           %Fraction of population kept
keep=floor(selection*popsize);         % #population members that survive
M=ceil((popsize-keep)/2);              % number of mating pair
odds=1;
for ii=2:keep
    odds=[odds ii*ones(1,ii)];
end
Nodds=length(odds);

%% Create the initial population
iga=0;                                 %Generation counter and random
population
for iz=1:popsize
    pop(iz,:)=randperm(fac);
end
cost=feval(ff,pop);                    %Calculate cost based on initial
population (Evaluate initital fitness)
[cost,ind]=sort(cost);
pop=pop(ind,:);
```

```
minc{1}=min(cost);
meanc{1}=mean(cost);

%% Entering the loop of crossover and mutation with stopping criteria
solstring=[];                          %Generating initial solution
vector for plotting
while iga<maxit
    iga=iga+1;
    pick1=ceil(Nodds*rand(1,M));       %parent #1
    pick2=ceil(Nodds*rand(1,M));       %parent #2
    ma=odds(pick1);            %ma and pa contain the indicies of the
parents
    pa=odds(pick2);
    for ic=1:M                 %Perform crossover CX operator (mating)
        mate1=pop(ma(ic),:);
        mate2=pop(pa(ic),:);
        indx=2*(ic-1)+1;       %Starting at one and skipping every other
one
        xp=ceil(rand*fac);  %Random value between 1 and Number of
facilities
        temp=mate1;
        x0=xp;
        while mate1(xp)~=temp(x0)
            mate1(xp)=mate2(xp);
            mate2(xp)=temp(xp);
            xs=find(temp==mate1(xp));
            xp=xs;
        end
        pop(keep+indx,:)=mate1;
        pop(keep+indx+1,:)=mate2;
    end
    nmut=ceil(popsize*fac*mutation); %Perform mutation operator
    for ic = 1:nmut
        row1=ceil(rand*(popsize-1))+1;
        col1=ceil(rand*fac);
        col2=ceil(rand*fac);
        temp=pop(row1,col1);
        pop(row1,col1)=pop(row1,col2);
        pop(row1,col2)=temp;
        im(ic)=row1;
    end
    cost=feval(ff,pop);            %Calculate pop by using objective
function
    part=pop; costt=cost;
    [cost,ind]=sort(cost);         %Sort the costs and associated parameters
    pop=pop(ind,:);                %Rearrange population matrix
    minc{iga}=min(cost);           %Calculating minimum cost
    meanc{iga}=mean(cost);         %Calculating mean cost
    solstring(iga)=cost(1);        %Inputting result of each iteration into
solution vector
    str=sort(solstring,'descend');   %Sorting all iterations' results
in descending order
    standarddev=std(str);          %Caculating standard deviation of all
iterations' results
    mstr=mean(str);                %Caculating mean of all iterations'
results
```

```
    finalcost=min(solstring); %Extracting minimum value of all
iterations
end %iga

%% Converting result in order to inputting into database
res='result.xlsx';
sheet = 1;
A = {num2str(pop(1,:))};
A = A{1,1};
A = strrep(A,'   ',',');
A = strrep(A,'  ',',');
D1 = strread(A,'%u','delimiter',',')';

%% Displaying Results
disp(['Best string layout: ' num2str(pop(1,:))])
disp(['Final Cost for layout = ' num2str(finalcost)])
disp(['Standard Deviation = ' num2str(standarddev)])
disp(['Mean of Data = ' num2str(mstr)])
disp(['Worst cost of layout = ' num2str(max(solstring))])
%plot(str)

%% Connecting to MySQL database and exporting
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Checking which phase is the site generation currently is on
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 4);
value = db.query();
value = cell2mat(struct2cell(value));

%% Slecting set of facility according to that current phase
db.prepareStatement('SELECT Facility FROM decision WHERE `Phase` =
"{Si}"', value);
sel_facility = db.query();
sel = cell2mat(struct2cell(sel_facility));
[t,r] = size(sel);
sel = reshape(sel,1,t);

%% Obtaining Frenquency matrix from big matrix according selected
facilities
fren = xlsread('duy.xlsx');
select = sel;                          %---------Modify from datbase-----
-----%
[~,n] = size(select);
for i = 1:n
    for x = 1:n
        j = select(i);
        y = select(x);
        acfren(i,x) = fren(j,y);
    end
end
[g,h] = size(acfren);
mainfren(1:g,1:g) = acfren;
```

```
%% Understanding which facility in the matrix is dummy
D2 = D1;
[c,d] = size(D2);
F = sum(mainfren);
for i = 1:d;
    if F(i) <= d - 1
        D2(D2 == i) = 0;
    else
    end;
end;
D1 = reshape(D1,z,1);
D2 = reshape(D2,z,1);
[m,n]=size(D2);                %Dimension of solution string
C = 1:m;
for i = 1:m
    a = i;
    b = D2(C == i);
    c = D1(C == i);
    db.prepareStatement('INSERT INTO layout (ID) VALUES ("{Si}")', a);
    db.query();
    db.prepareStatement('UPDATE layout SET FACILITY = "{Si}" WHERE ID =
"{Si}"', b, a);
    db.query();
    db.prepareStatement('UPDATE layout SET FACILITY2 = "{Si}" WHERE ID
= "{Si}"', c, a);
    db.query();
end;
db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE ID =
"{Si}"', 0, 1);
db.query();
db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE ID =
"{Si}"', 1, 3);
db.query();
db.close();
```

File slpfun1:

```
function tcost=slpfun1(pop)
clc
global x y

%% Connecting to MySQL Database
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));
mainfren = ones(z);
mainfren(logical(eye(size(mainfren)))) = 0;

%% Checking which phase is the site generation currently is on
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 4);
value = db.query();
value = cell2mat(struct2cell(value));

%% Selecting set of facility according to that current phase
db.prepareStatement('SELECT Facility FROM decision WHERE `Phase` =
"{Si}"', value);
sel_facility = db.query();
sel = cell2mat(struct2cell(sel_facility));
[t,r] = size(sel);
sel = reshape(sel,1,t);

%% Obtaining Frenquency matrix from big matrix according selected
facilities
fren = xlsread('duy.xlsx');
select = sel;                                    %---------Modify
from database----------%
[~,n] = size(select);
for i = 1:n
    for x = 1:n
        j = select(i);
        y = select(x);
        acfren(i,x) = fren(j,y);
    end
end
[g,h] = size(acfren);
mainfren(1:g,1:g) = acfren;

%% Extracting coordinates from MySQL Dtatabase according to selected
%% facilities
for i = 1:z
    db.prepareStatement('SELECT X FROM coordinates WHERE `ID` =
"{Si}"', i);
```

```
    X (i) = db.query();
    db.prepareStatement('SELECT Y FROM coordinates WHERE `ID` =
"{Si}"', i);
    Y (i) = db.query();
end
X = reshape(struct2cell(X),z,1);
Y = reshape(struct2cell(Y),z,1);
B = cell2mat([X Y]);

%% Computing Travel distance matrix based on selected coordinates
[m,n]=size(B);                          %Compute size of coordinates matrix
for i = 1:m                             %Loop starting & calculating
distance between locations
    for j = 1:m
      A(i,j) = sqrt((B(i,1) - B(j,1))^2 + (B(i,2) - B(j,2))^2);
    end;
end;
db.close();

%% Objective Function Calculation
[Npop,Nfac]=size(pop);                  %calculate the size of
population matrix
dist=A;                                 %Extract distance matrix from
excel file
fren = mainfren;                          %Extract frequency matrix from
excel file
for ir=1:Npop
    string1=pop(ir,:);                  %Extract each string (layout) in
number of population matrix
    string2=(1:m);                      %Location string
    [~,N]=size(string1);                %Obtaining size of string 1
    [~,L]=size(string2);                %Obtaining size of string 2
    P=zeros(N,L);                       %Generating zeros matrix based
on dimension of string1 & string2
    for ii=1:m
        P(string2(ii),string1(ii))=1;   %Replacing value of 1 in zeros
matrix to create permutation matrix from both string layout
        tcost=dist*fren*P;              %Calculating the total cost of
objective function (matrix)
        sol=sum(tcost(:));              %Calculating the total cost of
objective function (determinant of matrix)
    end
end
```

File slp2:

```
clear
clc
global iga x y
%% Connecting to MySQL and extract coordinates of available locations
%% generated from RTLS and C# code
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Deleting phase 1 information
db.prepareStatement('DELETE FROM decision WHERE `Phase` = "{Si}"', 1);
assert(~isequal(-1, db.query));

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));

%% Checking which phase is the site generation currently is on
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 4);
value1 = db.query();
value1 = cell2mat(struct2cell(value1));
value2 = value1 -1;

%% Selecting set of facility in previous phase with "fixed" status
db.prepareStatement('SELECT Facility FROM decision WHERE `Status` =
"{Si}"', value2);
phase = db.query();
sel = cell2mat(struct2cell(phase));
[t,r] = size(sel);
sel = reshape(sel,1,t);

%% Extracting previous solution layout
for i = 1:z
   db.prepareStatement('SELECT FACILITY2 FROM layout WHERE `ID` =
"{Si}"', i);
   T (i) = db.query();
end;
R = reshape(struct2cell(T),1,z);
last_sol = cell2mat(R);

%% Taking out location belong to fixed facilities
for i = 1:t
%Adjustable
    k = sel(i);
    last_sol(last_sol == k) = 0;
end;
last_sol = sort(last_sol);
for i = 1:z
    l = last_sol(i);
```

```
    db.prepareStatement('SELECT X FROM coordinates WHERE `ID` =
"{Si}"', 1);
    X (i) = db.query();
    db.prepareStatement('SELECT Y FROM coordinates WHERE `ID` =
"{Si}"', 1);
    Y (i) = db.query();
end;
X = reshape(struct2cell(X),z,1);
Y = reshape(struct2cell(Y),z,1);
B = cell2mat([X Y]);
[m,z] = size(B);

%% Selecting set of facility according to that current phase
db.prepareStatement('SELECT Facility FROM decision WHERE `Phase` =
"{Si}"', value1);
sel_facility = db.query();
sel2 = cell2mat(struct2cell(sel_facility));
[t,r] = size(sel2);
sel2 = reshape(sel2,1,t);
for i = 1:length(sel)
    y = sel(i);
    sel2(sel2 == y) = [];
end
db. close();

%% Obtaining Frenquency matrix from big matrix according selected
facilities
fren = xlsread('duy.xlsx');
select = sel2;                                    %---------Modify
from datbase----------%
[~,n] = size(select);
for i = 1:n
    for x = 1:n
        j = select(i);
        y = select(x);
        acfren(i,x) = fren(j,y);
    end
end
r = min(sel2);
g = max(sel2);
mainfren = ones(m);
mainfren(logical(eye(size(mainfren)))) = 0;
mainfren(r:g,r:g) = acfren;

%% Setup the GA Parameters
ff='slpfun2';                          %Objective function
fac=m;                                 %Number of facilities
loc=m;                                 %Number of locations

maxit=50;                              %Maximum number of iterations
popsize=m^2;                           %Population Size
mutation=0.01;                         %Set mutation rate
selection=1;                           %Fraction of population kept

keep=floor(selection*popsize);         % #population members that survive
M=ceil((popsize-keep)/2);              % number of mating pair
```

```
odds=1;
for ii=2:keep
    odds=[odds ii*ones(1,ii)];
end
Nodds=length(odds);

%% Create the initial population
iga=0;                                  %Generation counter and random
population
for iz=1:popsize
    pop(iz,:)=randperm(fac);
end
cost=feval(ff,pop);                     %Calculate cost based on initial
population (Evaluate initital fitness)
[cost,ind]=sort(cost);
pop=pop(ind,:);
minc{1}=min(cost);
meanc{1}=mean(cost);

%% Entering the loop of crossover and mutation with stopping criteria
solstring=[];                           %Generating initial solution
vector for plotting
while iga<maxit
    iga=iga+1;
    pick1=ceil(Nodds*rand(1,M));        %parent #1
    pick2=ceil(Nodds*rand(1,M));        %parent #2
    ma=odds(pick1);                     %ma and pa contain the indicies of
the parents
    pa=odds(pick2);
    for ic=1:M                          %Perform crossover CX operator
(mating)
        mate1=pop(ma(ic),:);
        mate2=pop(pa(ic),:);
        indx=2*(ic-1)+1;                %Starting at one and skipping
every other one
        xp=ceil(rand*fac);              %Random value between 1 and Number
of facilities
        temp=mate1;
        x0=xp;
        while mate1(xp)~=temp(x0)
            mate1(xp)=mate2(xp);
            mate2(xp)=temp(xp);
            xs=find(temp==mate1(xp));
            xp=xs;
        end
        pop(keep+indx,:)=mate1;
        pop(keep+indx+1,:)=mate2;
    end
    nmut=ceil(popsize*fac*mutation); %Perform mutation operator
    for ic = 1:nmut
        row1=ceil(rand*(popsize-1))+1;
        col1=ceil(rand*fac);
        col2=ceil(rand*fac);
        temp=pop(row1,col1);
        pop(row1,col1)=pop(row1,col2);
        pop(row1,col2)=temp;
```

```
        im(ic)=row1;
    end
    cost=feval(ff,pop);              %Calculate pop by using objective
function
    part=pop; costt=cost;
    [cost,ind]=sort(cost);           %Sort the costs and associated
parameters
    pop=pop(ind,:);                  %Rearrange population matrix
    minc{iga}=min(cost);             %Calculating minimum cost
    meanc{iga}=mean(cost);           %Calculating mean cost
    solstring(iga)=cost(1);          %Inputting result of each
iteration into solution vector
    str=sort(solstring,'descend');   %Sorting all iterations' results
in descending order
    standarddev=std(str);            %Caculating standard deviation of
all iterations' results
    mstr=mean(str);                  %Caculating mean of all
iterations' results
    finalcost=min(solstring);        %Extracting minimum value of all
iterations
end %iga

%% Converting result in order to inputting into database
res='result.xlsx';
sheet = 1;
A = {num2str(pop(1,:))};
A = A{1,1};
A = strrep(A,'    ',',');
A = strrep(A,'  ',',');
D1 = strread(A,'%u','delimiter',',')';

D2 = D1;
[~,d] = size(D2);
F = sum(mainfren);
for i = 1:d;
    if F(i) <= d - 1
        D2(D2 == i) = 0;
    else
    end;
end;

%% Displaying Results
disp(['Best string layout: ' num2str(pop(1,:))])
disp(['Final Cost for layout = ' num2str(finalcost)])
disp(['Standard Deviation = ' num2str(standarddev)])
disp(['Mean of Data = ' num2str(mstr)])
disp(['Worst cost of layout = ' num2str(max(solstring))])
%plot(str)

%% Connecting to MySQL database and exporting
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Obtaining maximum number of available locations on job site
```

```
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));

%% Extracting previous string solution from MySQL database
for i = 1:z
   db.prepareStatement('SELECT FACILITY FROM layout WHERE `ID` =
"{Si}"', i);
   L(i) = db.query();
end;
R = reshape(struct2cell(L),1,z);
prev_sol = cell2mat(R);

%% Selecting set of facility in previous phase with "fixed" status
db.prepareStatement('SELECT Facility FROM decision WHERE `Status` =
"{Si}"', value2);
phase = db.query();
sel = cell2mat(struct2cell(phase));
[t,r] = size(sel);
sel = reshape(sel,1,t);
sel = prev_sol(ismember(prev_sol,sel));

%% Adding fixed facilities into new solution string
[m,n] = size(sel);
for i = 1:n
    r = sel(i);
    sel_fac (i)= prev_sol(prev_sol == r);
end
fixed_fac = arrayfun(@(x) find(prev_sol == x,1,'first'), sel_fac);
new_sol = D2;
c = false(1,length(new_sol)+length(fixed_fac));
c(fixed_fac)=true;
result=nan(size(c));
result(~c)=new_sol;
result(c)= sel_fac;

%% Inputting updated solution string into MySQL Database
W = reshape(result,length(result),1);
[m,n]=size(W);
C = 1:m;
for i = 1:m
    a = i;
    b = W(C==i);
    db.prepareStatement('INSERT INTO layout (ID) VALUES ("{Si}")', a);
    db.query();
    db.prepareStatement('UPDATE layout SET FACILITY = "{Si}" WHERE ID =
"{Si}"', b, a);
    db.query();
end;

%% Adding fixed facilities into new solution string -------- RAW
[~,n] = size(sel);
for ii = 1:length(D1)
    y = D1(ii);
    if y > min(sel)
```

```
        D1 (ii) = y + n;
    else
    end
end
for i = 1:n
    r = sel(i);
    sel_fac (i)= prev_sol(prev_sol == r);
end
fixed_fac = arrayfun(@(x) find(prev_sol == x,1,'first'), sel_fac);
new_sol = D1;
c = false(1,length(new_sol)+length(fixed_fac));
c(fixed_fac)=true;
result1=nan(size(c));
result1(~c)=new_sol;
result1(c)= sel_fac;

%% Inputting updated solution string into MySQL Database -------- RAW
W = reshape(result1,length(result1),1);
[m,n]=size(W);
C = 1:m;
for i = 1:m
    a = i;
    b = W(C==i);
    db.prepareStatement('INSERT INTO layout (ID) VALUES ("{Si}")', a);
    db.query();
    db.prepareStatement('UPDATE layout SET FACILITY2 = "{Si}" WHERE ID
= "{Si}"', b, a);
    db.query();
end;

%% Updating triggertable in MySQL database for BIM
db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE ID =
"{Si}"', 0, 1);
db.query();
db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE ID =
"{Si}"', 1, 3);
db.query();
db.close();
```

File slpfun2:

```
function tcost=slpfun2(pop)
clc
global x y

%% Connecting to MySQL Database
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Deleting phase 1 information
db.prepareStatement('DELETE FROM decision WHERE `Phase` = "{Si}"', 1);
assert(~isequal(-1, db.query));

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));

%% Checking which phase is the site generation currently is on
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 4);
value1 = db.query();
value1 = cell2mat(struct2cell(value1));
value2 = value1 -1;

%% Selecting set of facility in previous phase with "fixed" status
db.prepareStatement('SELECT Facility FROM decision WHERE `Status` =
"{Si}"', value2);
phase = db.query();
sel = cell2mat(struct2cell(phase));
[t,r] = size(sel);
sel = reshape(sel,1,t);

%% Extracting previous solution layout
for i = 1:z
   db.prepareStatement('SELECT FACILITY2 FROM layout WHERE `ID` =
"{Si}"', i);
   T (i) = db.query();
end;
R = reshape(struct2cell(T),1,z);
prev_sol = cell2mat(R);

%% Taking out location belong to fixed facilities
for i = 1:t
%Adjustable
    k = sel(i);
    prev_sol(prev_sol == k) = 0;
end;
prev_sol = sort(prev_sol);
for i = 1:z
    l = prev_sol(i);
```

```
    db.prepareStatement('SELECT X FROM coordinates WHERE `ID` =
"{Si}"', 1);
    X (i) = db.query();
    db.prepareStatement('SELECT Y FROM coordinates WHERE `ID` =
"{Si}"', 1);
    Y (i) = db.query();
end;
X = reshape(struct2cell(X),z,1);
Y = reshape(struct2cell(Y),z,1);
B = cell2mat([X Y]);
[u,z] = size(B);

%% Selecting set of facility according to that current phase
db.prepareStatement('SELECT Facility FROM decision WHERE `Phase` =
"{Si}"', value1);
sel_facility = db.query();
sel2 = cell2mat(struct2cell(sel_facility));
[t,r] = size(sel2);
sel2 = reshape(sel2,1,t);
for i = 1:length(sel)
    y = sel(i);
    sel2(sel2 == y) = [];
end

%% Obtaining Frenquency matrix from big matrix according selected
facilities
fren = xlsread('duy.xlsx');
select = sel2;                                          %---------Modify
from datbase----------%
[~,n] = size(select);
for i = 1:n
    for x = 1:n
        j = select(i);
        y = select(x);
        acfren(i,x) = fren(j,y);
    end
end
r = min(sel2);
g = max(sel2);
mainfren = ones(u);
mainfren(logical(eye(size(mainfren)))) = 0;
mainfren(r:g,r:g) = acfren;

%% Computing Travel distance matrix based on selected coordinates
[m,n]=size(B);                          %Compute size of coordinates matrix
for i = 1:m                             %Loop starting & calculating
distance between locations
    for j = 1:m
      A(i,j) = sqrt((B(i,1) - B(j,1))^2 + (B(i,2) - B(j,2))^2);
    end;
end;
db.close();

%% Objective Function Calculation
[Npop,Nfac]=size(pop);                  %calculate the size of
population matrix
```

```
dist=A;                                    %Extract distance matrix from
excel file
fren = mainfren;                             %Extract frequency matrix from
excel file
for ir=1:Npop
    string1=pop(ir,:);                     %Extract each string (layout) in
number of population matrix
    string2=(1:m);                         %Location string
    [~,N]=size(string1);                   %Obtaining size of string 1
    [~,L]=size(string2);                   %Obtaining size of string 2
    P=zeros(N,L);                          %Generating zeros matrix based
on dimension of string1 & string2
    for ii=1:m
        P(string2(ii),string1(ii))=1;      %Replacing value of 1 in zeros
matrix to create permutation matrix from both string layout
        tcost=dist*fren*P;                 %Calculating the total cost of
objective function (matrix)
        sol=sum(tcost(:));                 %Calculating the total cost of
objective function (determinant of matrix)
    end
end
```

File slp3:

```
clear
clc
global iga x y
%% Connecting to MySQL Database
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Deleting phase 2 information
db.prepareStatement('DELETE FROM decision WHERE `Phase` = "{Si}"', 2);
assert(~isequal(-1, db.query));

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));

%% Checking which phase is the site generation currently is on
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 4);
value1 = db.query();
value1 = cell2mat(struct2cell(value1));
value2 = value1 -2;

%% Selecting set of facility in previous phase with "fixed" status
db.prepareStatement('SELECT Facility FROM decision WHERE `Status` =
"{Si}"', value2);
phase = db.query();
sel = cell2mat(struct2cell(phase));
[t,r] = size(sel);
sel = reshape(sel,1,t);
[a,b] = size(sel);

%% Extracting previous solution layout
for i = 1:z
   db.prepareStatement('SELECT FACILITY2 FROM layout WHERE `ID` =
"{Si}"', i);
   T (i) = db.query();
end;
R = reshape(struct2cell(T),1,z);
prev_sol = cell2mat(R);

%% Taking out location belong to fixed facilities
for i = 1:t
%Adjustable
    k = sel(i);
    prev_sol(prev_sol == k) = 0;
end;
prev_sol = sort(prev_sol);
for i = 1:z
    l = prev_sol(i);
```

```
    db.prepareStatement('SELECT X FROM coordinates WHERE `ID` =
"{Si}"', 1);
    X (i) = db.query();
    db.prepareStatement('SELECT Y FROM coordinates WHERE `ID` =
"{Si}"', 1);
    Y (i) = db.query();
end;
X = reshape(struct2cell(X),z,1);
Y = reshape(struct2cell(Y),z,1);
B = cell2mat([X Y]);
[u,y] = size(B);

%% Selecting set of facility according to that current phase
db.prepareStatement('SELECT Facility FROM decision WHERE `Phase` =
"{Si}"', value1);
sel_facility = db.query();
sel2 = cell2mat(struct2cell(sel_facility));
[t,r] = size(sel2);
sel2 = reshape(sel2,1,t);
for i = 1:length(sel)
    y = sel(i);
    sel2(sel2 == y) = [];
end

%% Obtaining Frenquency matrix from big matrix according selected
facilities
fren = xlsread('duy.xlsx');
select = sel2;                                    %---------Modify
from datbase----------%
[~,n] = size(select);
for i = 1:n
    for x = 1:n
        j = select(i);
        y = select(x);
        acfren(i,x) = fren(j,y);
    end
end
r = min(sel2);
g = max(sel2);
mainfren = ones(z);
mainfren(logical(eye(size(mainfren)))) = 0;
mainfren(r:g,r:g) = acfren;
mainfren = mainfren(b+1:z,b+1:z);
[m,n]=size(B);                    %Compute size of coordinates matrix

%% Setup the GA Parameters
ff='slpfun3';                      %Objective function
fac=m;                            %Number of facilities
loc=m;                            %Number of locations

maxit=50;                         %Maximum number of iterations
popsize=m^2;                      %Population Size
mutation=0.01;                    %Set mutation rate
selection=1;                      %Fraction of population kept

keep=floor(selection*popsize);    % #population members that survive
```

```
M=ceil((popsize-keep)/2);              % number of mating pair
odds=1;
for ii=2:keep
    odds=[odds ii*ones(1,ii)];
end
Nodds=length(odds);

%% Create the initial population
iga=0;                                 %Generation counter and random
population
for iz=1:popsize
    pop(iz,:)=randperm(fac);
end
cost=feval(ff,pop);                    %Calculate cost based on initial
population (Evaluate initital fitness)
[cost,ind]=sort(cost);
pop=pop(ind,:);
minc{1}=min(cost);
meanc{1}=mean(cost);

%% Entering the loop of crossover and mutation with stopping criteria
solstring=[];                          %Generating initial solution
vector for plotting
while iga<maxit
    iga=iga+1;
    pick1=ceil(Nodds*rand(1,M));       %parent #1
    pick2=ceil(Nodds*rand(1,M));       %parent #2
    ma=odds(pick1);                    %ma and pa contain the indicies of
the parents
    pa=odds(pick2);
    for ic=1:M                         %Perform crossover CX operator
(mating)
        mate1=pop(ma(ic),:);
        mate2=pop(pa(ic),:);
        indx=2*(ic-1)+1;               %Starting at one and skipping
every other one
        xp=ceil(rand*fac);             %Random value between 1 and Number
of facilities
        temp=mate1;
        x0=xp;
        while mate1(xp)~=temp(x0)
            mate1(xp)=mate2(xp);
            mate2(xp)=temp(xp);
            xs=find(temp==mate1(xp));
            xp=xs;
        end
        pop(keep+indx,:)=mate1;
        pop(keep+indx+1,:)=mate2;
    end
    nmut=ceil(popsize*fac*mutation); %Perform mutation operator
    for ic = 1:nmut
        row1=ceil(rand*(popsize-1))+1;
        col1=ceil(rand*fac);
        col2=ceil(rand*fac);
        temp=pop(row1,col1);
        pop(row1,col1)=pop(row1,col2);
```

```
        pop(row1,col2)=temp;
        im(ic)=row1;
    end
    cost=feval(ff,pop);              %Calculate pop by using objective
function
    part=pop; costt=cost;
    [cost,ind]=sort(cost);           %Sort the costs and associated
parameters
    pop=pop(ind,:);                  %Rearrange population matrix
    minc{iga}=min(cost);             %Calculating minimum cost
    meanc{iga}=mean(cost);           %Calculating mean cost
    solstring(iga)=cost(1);          %Inputting result of each
iteration into solution vector
    str=sort(solstring,'descend');   %Sorting all iterations' results
in descending order
    standarddev=std(str);            %Caculating standard deviation of
all iterations' results
    mstr=mean(str);                  %Caculating mean of all
iterations' results
    finalcost=min(solstring);        %Extracting minimum value of all
iterations
end %iga

%% Converting result in order to inputting into database
res='result.xlsx';
sheet = 1;
A = {num2str(pop(1,:))};
A = A{1,1};
A = strrep(A,'    ',',');
A = strrep(A,'  ',',');
D1 = strread(A,'%u','delimiter',',')';

D2 = D1;
[~,d] = size(D2);
F = sum(mainfren);
for i = 1:d;
    if F(i) <= d - 1
        D2(D2 == i) = 0;
    else
    end;
end;
for i = 1:length(D2)
    r = D2(i);
    if r > 0
        D2 (i) = r + length(sel);
    else
    end
end

%% Displaying Results
disp(['Best string layout: ' num2str(pop(1,:))])
disp(['Final Cost for layout = ' num2str(finalcost)])
disp(['Standard Deviation = ' num2str(standarddev)])
disp(['Mean of Data = ' num2str(mstr)])
disp(['Worst cost of layout = ' num2str(max(solstring))])
%plot(str)
```

```
%% Connecting to MySQL database and exporting
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));

%% Extracting previous string solution from MySQL database
for i = 1:z
   db.prepareStatement('SELECT FACILITY FROM layout WHERE `ID` =
"{Si}"', i);
   L(i) = db.query();
end;
R = reshape(struct2cell(L),1,z);
prev_sol = cell2mat(R);

%% Selecting set of facility in previous phase with "fixed" status
db.prepareStatement('SELECT Facility FROM decision WHERE `Status` =
"{Si}"', value2);
phase = db.query();
sel = cell2mat(struct2cell(phase));
[t,r] = size(sel);
sel = reshape(sel,1,t);
sel = prev_sol(ismember(prev_sol,sel));

%% Adding fixed facilities into new solution string
[m,n] = size(sel);
for i = 1:n
    r = sel(i);
    sel_fac (i)= prev_sol(prev_sol == r);
end
fixed_fac = arrayfun(@(x) find(prev_sol == x,1,'first'), sel_fac);
new_sol = D2;
c = false(1,length(new_sol)+length(fixed_fac));
c(fixed_fac)=true;
result=nan(size(c));
result(~c)=new_sol;
result(c)= sel_fac;

%% Inputting updated solution string into MySQL Database
W = reshape(result,length(result),1);
[m,n]=size(W);
C = 1:m;
for i = 1:m
    a = i;
    b = W(C==i);
    db.prepareStatement('INSERT INTO layout (ID) VALUES ("{Si}")', a);
    db.query();
    db.prepareStatement('UPDATE layout SET FACILITY = "{Si}" WHERE ID =
"{Si}"', b, a);
```

```
    db.query();
end;

%% Adding fixed facilities into new solution string -------- RAW
[~,n] = size(sel);
for ii = 1:length(D1)
    y = D1(ii);
    if y > min(sel)
        D1 (ii) = y + n;
    else
    end
end
for i = 1:n
    r = sel(i);
    sel_fac (i)= prev_sol(prev_sol == r);
end
fixed_fac = arrayfun(@(x) find(prev_sol == x,1,'first'), sel_fac);
new_sol = D1;
c = false(1,length(new_sol)+length(fixed_fac));
c(fixed_fac)=true;
result1=nan(size(c));
result1(~c)=new_sol;
result1(c)= sel_fac;

%% Inputting updated solution string into MySQL Database -------- RAW
W = reshape(result1,length(result1),1);
[m,n]=size(W);
C = 1:m;
for i = 1:m
    a = i;
    b = W(C==i);
    db.prepareStatement('INSERT INTO layout (ID) VALUES ("{Si}")', a);
    db.query();
    db.prepareStatement('UPDATE layout SET FACILITY2 = "{Si}" WHERE ID
= "{Si}"', b, a);
    db.query();
end;

%% Updating triggertable in MySQL database for BIM
db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE ID =
"{Si}"', 0, 1);
db.query();
db.prepareStatement('UPDATE triggertable SET VALUE = "{Si}" WHERE ID =
"{Si}"', 1, 3);
db.query();
db.close();
```

File slpfun3:

```
function tcost=slpfun3(pop)
clc
global x y

%% Connecting to MySQL Database
addpath(fullfile(pwd, 'src'));
javaaddpath('mysql-connector-java-5.1.6-bin.jar');
import MySQLDatabase;
db = MySQLDatabase('localhost', 'SiteLayout', 'root', 'hammad');

%% Deleting phase 2 information
db.prepareStatement('DELETE FROM decision WHERE `Phase` = "{Si}"', 2);
assert(~isequal(-1, db.query));

%% Obtaining maximum number of available locations on job site
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 2);
z = db.query();
z = cell2mat(struct2cell(z));

%% Checking which phase is the site generation currently is on
db.prepareStatement('SELECT Value FROM triggertable WHERE `ID` =
"{Si}"', 4);
value1 = db.query();
value1 = cell2mat(struct2cell(value1));
value2 = value1 -2;

%% Selecting set of facility in previous phase with "fixed" status
db.prepareStatement('SELECT Facility FROM decision WHERE `Status` =
"{Si}"', value2);
phase = db.query();
sel = cell2mat(struct2cell(phase));
[t,r] = size(sel);
sel = reshape(sel,1,t);
[a,b] = size(sel);

%% Extracting previous solution layout
for i = 1:z
   db.prepareStatement('SELECT FACILITY2 FROM layout WHERE `ID` =
"{Si}"', i);
   T (i) = db.query();
end;
R = reshape(struct2cell(T),1,z);
prev_sol = cell2mat(R);

%% Taking out location belong to fixed facilities
for i = 1:t
%Adjustable
    k = sel(i);
    prev_sol(prev_sol == k) = 0;
end;
prev_sol = sort(prev_sol);
for i = 1:z
```

```
    l = prev_sol(i);
    db.prepareStatement('SELECT X FROM coordinates WHERE `ID` =
"{Si}"', l);
    X (i) = db.query();
    db.prepareStatement('SELECT Y FROM coordinates WHERE `ID` =
"{Si}"', l);
    Y (i) = db.query();
end;
X = reshape(struct2cell(X),z,1);
Y = reshape(struct2cell(Y),z,1);
B = cell2mat([X Y]);
[u,y] = size(B);

%% Selecting set of facility according to that current phase
db.prepareStatement('SELECT Facility FROM decision WHERE `Phase` =
"{Si}"', value1);
sel_facility = db.query();
sel2 = cell2mat(struct2cell(sel_facility));
[t,r] = size(sel2);
sel2 = reshape(sel2,1,t);
for i = 1:length(sel)
    y = sel(i);
    sel2(sel2 == y) = [];
end

%% Obtaining Frenquency matrix from big matrix according selected
facilities
fren = xlsread('duy.xlsx');
select = sel2;                                          %---------Modify
from datbase----------%
[~,n] = size(select);
for i = 1:n
    for x = 1:n
        j = select(i);
        y = select(x);
        acfren(i,x) = fren(j,y);
    end
end
r = min(sel2);
g = max(sel2);
mainfren = ones(z);
mainfren(logical(eye(size(mainfren)))) = 0;
mainfren(r:g,r:g) = acfren;
mainfren = mainfren(b+1:z,b+1:z);

%% Computing Travel distance matrix based on selected coordinates
[m,n]=size(B);                      %Compute size of coordinates matrix
for i = 1:m                         %Loop starting & calculating
distance between locations
    for j = 1:m
      A(i,j) = sqrt((B(i,1) - B(j,1))^2 + (B(i,2) - B(j,2))^2);
    end;
end;
db.close();

%% Objective Function Calculation
```

```
[Npop,Nfac]=size(pop);                    %calculate the size of
population matrix
dist=A;                                   %Extract distance matrix from
excel file
fren = mainfren;                            %Extract frequency matrix from
excel file
for ir=1:Npop
    string1=pop(ir,:);                    %Extract each string (layout) in
number of population matrix
    string2=(1:m);                        %Location string
    [~,N]=size(string1);                  %Obtaining size of string 1
    [~,L]=size(string2);                  %Obtaining size of string 2
    P=zeros(N,L);                         %Generating zeros matrix based
on dimension of string1 & string2
    for ii=1:m
        P(string2(ii),string1(ii))=1;     %Replacing value of 1 in zeros
matrix to create permutation matrix from both string layout
        tcost=dist*fren*P;                %Calculating the total cost of
objective function (matrix)
        sol=sum(tcost(:));                %Calculating the total cost of
objective function (determinant of matrix)
    end
end
```

C# Code

RTLS Program:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Net;
using System.Reflection;

using CSL.RTLS;
using CSL.RTLS.Events;
using CSL.RTLS.Packets;
using CSL.RTLS.Devices;
using CSL.RTLS.Anchors;
using CSL.RTLS.Tags;
using CSL.RTLS.Cells;
using CSL.RTLS.Filters;
using CSL.RTLS.Engines;
using CSL.RTLS.Structs;
using CSL.RTLS.Products.SpecV01;
using CSL.RTLS.Databases.DataSets;
using Architectures = CSL.RTLS.Architectures;
using RTLSProgram.Controls;

//hammad addition
using System.Linq;
using RTLSProgram.MultipathCharacterization;
using MySql.Data;
using MySql.Data.MySqlClient;
//hammad addition


namespace RTLSProgram
{
  public partial class FrmMain : Form
  {
            private struct Circle
            {
                  public PointF center;
                  public float radius;

                  public Circle(PointF c, float r)
                  {
                        center = c;
                        radius = r;
                  }
            }

            private enum CircleMode { None, All, Used }

            private RTLS rtls;
```

```
    private FrmAnchors frmAnchors;
    private FrmTags frmTags;

            private bool drawFloorMap;
            private bool drawBoundary;
            private bool drawGrid;
            private bool drawDiagonal;
            private bool drawRegion;
            private bool drawOneTagOnly;
            private bool drawTagActualLocation;
            private RangingCircle.Mode drawCircle;
            private bool closing = false;
            private bool anchor_register_flag = true;
            private ulong anchor_register_id = 0;
            private int activeCellPanelIndex;

            private FormWindowState wndState = FormWindowState.Normal;

            private Queue<DeviceEventArgs> rtls_queue = new
Queue<DeviceEventArgs>();
            private Queue<string> log_queue = new Queue<string>();
            private Queue<RegisterRequest> anchor_register_queue = new
Queue<RegisterRequest>();
            private Queue<RegisterRequest> tag_register_queue = new
Queue<RegisterRequest>();

            //private List<TagUserData> rtls.TagList = new List<TagUserData>();
            private List<Circle> circlesAll = new List<Circle>();
            private List<Circle> circlesUsed = new List<Circle>();
            public List<CellPanel> cellPanel_List = new List<CellPanel>();
            private CellGroupPanel cellGroupPanel;

      //hammad's addition
        public CellListDataSource cell_list_source;
        public BindingSource bsCellList;
        public List<RegionalCharacterization> regionalCharacterizeTable;


    public FrmMain(RTLS rtls)
    {
                this.rtls = rtls;

          regionalCharacterizeTable = new List<RegionalCharacterization>();
//hammad

                InitializeComponent();

                pnlLegend.DrawPanel = new
RTLSProgram.Controls.Panel.DrawPanelItems(pnlLegendPaint);
                pnlMaps.Anchor = AnchorStyles.Bottom | AnchorStyles.Left |
AnchorStyles.Right | AnchorStyles.Top;
    }

            private void FrmMain_Load(object sender, EventArgs e)
            {
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x1)
```

```
                        menuItemCell_1x1.Checked = true;
                    else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x2)
                        menuItemCell_1x2.Checked = true;
                    else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x3)
                        menuItemCell_1x3.Checked = true;
                    else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x4)
                        menuItemCell_1x4.Checked = true;
                    else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._2x2)
                        menuItemCell_2x2.Checked = true;
                    else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        menuItemCell_Global.Checked = true;
                    SetCellPanels();

                    wndState = this.WindowState;

                    drawFloorMap = false;
                    drawBoundary = true;
                    drawDiagonal = false;
                    drawGrid = true;
                    drawRegion = false;
                    drawCircle = RangingCircle.Mode.None;
                    drawOneTagOnly = false;
                    drawTagActualLocation = false;

                    this.Icon = RTLS.Icon;
                    this.Text = RTLS.Title + "Main Panel";
                    //this.Text = "CSL RTLS System v(" + RTLS.GetSystemVersion() +
" : Main Panel";

                    if (rtls.RTLSAssistant.frame_data_recording)
                    {
                        rbtnRecordFrameDataOn.ForeColor = Color.Red;
                        Font font = rbtnRecordFrameDataOn.Font;
                        font = new Font(font.FontFamily, font.Size,
FontStyle.Bold);
                        rbtnRecordFrameDataOn.Font = font;
                        rbtnRecordFrameDataOn.Checked = true;
                        rbtnRecordFrameDataOff.Checked = false;
                    }

                    //rtls.RTLSystem.AddRemoveHandler += OnAddRemove;
                    rtls.RTLSystem.DeviceNotifyHandler += OnDeviceNotify;
                    rtls.RTLSystem.DevicePacketHandler += OnDevicePacket;
                    rtls.RTLSAssistant.NotifyEventHandler += OnAssistantNotify;

                    //if (rtls.RTLSystem.Status != RTLSystem.SystemStatus.Running)
                    //    rtls.StartSystem();

                    tmrRTLS.Tick += (ss, ee) =>
                        {
                            rtls.DoLogging();
                        };
                    tmrRTLS.Enabled = true;
```

```
                        tmr500ms.Enabled = true;
                        tmr5000ms.Enabled = true;

                        frmAnchors = new FrmAnchors(this, rtls);
                        frmAnchors.Show();

                        frmTags = new FrmTags(rtls);
                        frmTags.Show();

                        if (rtls.CellList != null)
                                if (rtls.CellList.Count > 0)
                                        DisplayCellOnPanel(rtls.CellList[0]);

                        foreach (AnchorUserData aud in rtls.AnchorList)
                        {
                                if (aud.cell != null && aud.cell.masterAnchor != null
&& aud.health != null)
                                        if (aud.health.status ==
DeviceHealthStatus.HealthStatus.Idle)
                                                AddSystemMessage(new SystemMessage(rtls,
aud.cell.masterAnchor, aud.anchor,
SystemMessage.MessageType.SlaveAnchor_No_Response, "", DateTime.Now), false);
                        }

                        lbTime.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
                }

                private void FrmMain_FormClosing(object sender, FormClosingEventArgs
e)
                {
#if false
                        if(frameLog != null)
                                switch (MessageBox.Show(this, "Frame Data Recording!
Save data before exit?", Text, MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button1))
                                {
                                        case DialogResult.Yes:
                                                rbtnRecordFrameDataOff.Checked = true;
                                                break;
                                        case DialogResult.No:
                                                frameLog.Flush();
                                                frameLog.Dispose();
                                                frameLog = null;
                                                break;
                                        case DialogResult.Cancel:
                                                e.Cancel = true;
                                                return;
                                                break;
                                }
#endif

                        rtls.Config.Save();

                        closing = true;

                        Application.DoEvents();

                        //rtls.RTLSystem.AddRemoveHandler -= OnAddRemove;
```

```
                rtls.RTLSystem.DeviceNotifyHandler -= OnDeviceNotify;
                rtls.RTLSystem.DevicePacketHandler -= OnDevicePacket;
                rtls.RTLSAssistant.NotifyEventHandler -= OnAssistantNotify;

                tmr5000ms.Enabled = false;
                tmr500ms.Enabled = false;
                tmrRTLS.Enabled = false;
                tmrRTLS_Tick(this, null);

                if (frmAnchors != null)
                {
                        frmAnchors.Close();
                        frmAnchors = null;
                }
                if (frmTags != null)
                {
                        frmTags.Close();
                        frmTags = null;
                }

#if NanoLoc
                if (rtls.NanoLocSystem != null)
                {
                        rtls.NanoLocSystem.Stop((ex) =>
                                {
                                        rtls.RTLSystem.LogException(ex);
                                        MessageBox.Show(this, ex.Message, Text);
                                });
                        rtls.NanoLocSystem = null;
                }
#endif

                //rtls.StopSystem();
                foreach (Form frm in OwnedForms)
                {
                        frm.Close();
                }
        }

        private void tmrRTLS_Tick(object sender, EventArgs e)
        {
                rtls.RTLSAssistant.DoQueue(this);
                //rtls.RTLSAssistant.DoDelaySendMasterConfig();
    while (rtls_queue.Count > 0)
    {
        DeviceEventArgs de;
        lock (rtls_queue)
        {
            de = rtls_queue.Dequeue();
        }

        if (de is AddRemoveEventArgs)
            OnAddRemove(de as AddRemoveEventArgs);
        else if (de is DeviceNotifyEventArgs)
            OnDeviceNotify(de as DeviceNotifyEventArgs);
        else if (de is DevicePacketEventArgs)
            OnDevicePacket(de as DevicePacketEventArgs);
        else if (de is RTLSAssistant.DeviceStateChangeEventArgs)
```

```
                {
                    if (de.device is Anchor)
                        AnchorStateChange(de.device as Anchor);
                    else if (de.device is Tag)
                        TagStateChange(de.device as Tag);
                }
                else if (de is RTLSAssistant.DeviceRegisterRequestEventArgs)
                    DeviceRegisterRequest((de as
RTLSAssistant.DeviceRegisterRequestEventArgs).request);
                else if (de is RTLSAssistant.TagRegisterReaffirmEventArgs)
                    TagRegisterReaffirm(de as
RTLSAssistant.TagRegisterReaffirmEventArgs);
                else if (de is RTLSAssistant.TagDeregisterEventArgs)
                    TagDeregister(de as RTLSAssistant.TagDeregisterEventArgs);
                else if (de is RTLSAssistant.DeviceOldFirmwareVersionEventArgs)
                {
                    //OnDeviceOldFirmwareVersion(de as
RTLSAssistant.DeviceOldFirmwareVersionEventArgs);
                }
                else if (de is RTLSAssistant.DeviceRoamingEventArgs)
                    DeviceRoaming(de as RTLSAssistant.DeviceRoamingEventArgs);
                else if (de is RTLSAssistant.SystemMessageEventArgs)
                    FrmSystemMessage.Add(this, (de as
RTLSAssistant.SystemMessageEventArgs).system_message, false);
                else if (de is RTLSAssistant.FrameDataRecordEventArgs)
                    FrameDataRecord(de as RTLSAssistant.FrameDataRecordEventArgs);
            }
            while (log_queue.Count > 0)
            {
                string str;
                lock (log_queue)
                {
                    str = log_queue.Dequeue();
                }
                AddLog(str);
            }

            if (anchor_register_queue.Count > 0)
            {
                if (anchor_register_flag && !rtls.installation_wizard)
                {
                    anchor_register_flag = false;
                    AnchorRegisterRequest();
                }
            }

            lbTime.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
              }

              private void tmr500ms_Tick(object sender, EventArgs e)
              {
                     bool refresh = false;
                     List<CellUserDataV02> refresh_cell_list = new
List<CellUserDataV02>();
                     foreach (TagUserData t in rtls.TagList)
                     {
                            bool refresh_tag = false;
```

```
                        if (t.tag == null)
                                continue;
                        if (t.cell == null)
                                continue;

                        bool expRange = false;
                        bool expValidRange = false;
                        bool expPosition = false;

                        //if (t.lastRangingCycle == 0)
                                //continue;

                        if (rtls.Config.TagModeChangingCycle > 0)
                        {
                                if (t.cell.cycleCount - t.lastRangingCycle >
(ulong)rtls.Config.TagModeChangingCycle || t.lastRangingCycle == 0 &&
t.lastRanging == DateTime.MinValue)
                                        expRange = true;
                                if (t.cell.cycleCount - t.lastValidRangingCycle
> (ulong)rtls.Config.TagModeChangingCycle || t.lastValidRangingCycle == 0 &&
t.lastValidRanging == DateTime.MinValue)
                                        expValidRange = true;
                                if (t.cell.cycleCount - t.lastPositionCycle >
(ulong)rtls.Config.TagModeChangingCycle || t.lastPositionCycle == 0 &&
t.lastPosition == DateTime.MinValue)
                                        expPosition = true;
                        }
                        bool disconnected = !t.cell.connected;

#if false
                        int expireTime = t.cell.cfg.CalCycleTime() *
rtls.Config.TagModeChangingCycle;
                        DateTime now = DateTime.Now;
                        if (now.Subtract(t.lastRanging).TotalMilliseconds >
expireTime)
                                expRange = true;
                        if (now.Subtract(t.lastValidRanging).TotalMilliseconds
> expireTime)
                                expValidRange = true;
                        if (now.Subtract(t.lastPosition).TotalMilliseconds >
expireTime)
                                expPosition = true;
#endif

                        TagUserData.TagMode newMode = t.mode;
                        if (!disconnected)
                        {
                                if (!expPosition)
                                        newMode = TagUserData.TagMode.Active;
                                else
                                {
                                        if (!expValidRange)
                                                newMode =
TagUserData.TagMode.UnLocated;
                                        else if (!expRange)
                                        {
                                                newMode =
TagUserData.TagMode.OutOfReach;
```

```
#if false
                                                if
(!t.cell.cfg.NoMotionContinuousRanging || t.cell.cfg.MotionNoRanging)
                                                {
                                                        if (t.tag is
IMotionStatusDevice)
                                                        {
                                                                IMotionStatusDevice
motion_device = t.tag as IMotionStatusDevice;
                                                                if
(!t.cell.cfg.NoMotionContinuousRanging && motion_device.MotionStatus ==
MotionStatus.Stationary)
                                                                        newMode =
TagUserData.TagMode.NoMotionSleep;
                                                                else if
(t.cell.cfg.MotionNoRanging && motion_device.MotionStatus == MotionStatus.Moving)
                                                                        newMode =
TagUserData.TagMode.InMotionSleep;
                                                        }
                                                }
#endif
                                        }
                                        else
                                        {
                                                if
(t.cell.cfg.TagWakeupRegistration && (t.mode == TagUserData.TagMode.InMotionSleep
|| t.mode == TagUserData.TagMode.NoMotionSleep))
                                                {
                                                }
                                                else
                                                        newMode =
TagUserData.TagMode.Inactive;
                                        }
                                }

                                if (newMode != TagUserData.TagMode.Inactive)
                                {
                                        bool motion_sensor_sleep_enabled =
!t.cell.cfg.NoMotionContinuousRanging || t.cell.cfg.MotionNoRanging;
                                        if (motion_sensor_sleep_enabled)
                                        {
                                                if
(t.motion_sensor_sleep_trigger_handler.isTriggerCounting)
                                                {
                                                        if
(!t.cell.cfg.NoMotionContinuousRanging &&
t.motion_sensor_sleep_trigger_handler.motion_status == MotionStatus.Stationary)
                                                                newMode =
TagUserData.TagMode.TransitingNoMotionSleep;
                                                        else if
(t.cell.cfg.MotionNoRanging && t.motion_sensor_sleep_trigger_handler.motion_status
== MotionStatus.Moving)
                                                                newMode =
TagUserData.TagMode.TransitingInMotionSleep;
                                                }
                                                else if
(t.motion_sensor_sleep_trigger_handler.isLastTriggerArrive ||
```

```
t.motion_sensor_sleep_trigger_handler.isTriggerTimeOut || newMode ==
TagUserData.TagMode.OutOfReach)
                                                {
                                                        if
(!t.cell.cfg.NoMotionContinuousRanging &&
t.motion_sensor_sleep_trigger_handler.motion_status == MotionStatus.Stationary)
                                                                newMode =
TagUserData.TagMode.NoMotionSleep;
                                                        else if
(t.cell.cfg.MotionNoRanging && t.motion_sensor_sleep_trigger_handler.motion_status
== MotionStatus.Moving)
                                                                newMode =
TagUserData.TagMode.InMotionSleep;
                                                }
                                        }
                                }
                        }
                        else
                                newMode = TagUserData.TagMode.Inactive;

                        if (t.mode != newMode)
                        {
                                TagUserData.TagMode oldMode = t.mode;

                                t.mode = newMode;
                                refresh_tag = true;

                                string oldModeColor = "";
                                string newModeColor = "";

                                switch (oldMode)
                                {
                                        case TagUserData.TagMode.Active:
                                                oldModeColor = "BLUE";
                                                break;
                                        case
TagUserData.TagMode.TransitingInMotionSleep:
                                                oldModeColor = "ORANGE";
                                                break;
                                        case TagUserData.TagMode.InMotionSleep:
                                                oldModeColor = "BROWN";
                                                break;
                                        case
TagUserData.TagMode.TransitingNoMotionSleep:
                                                oldModeColor = "LIME";
                                                break;
                                        case TagUserData.TagMode.NoMotionSleep:
                                                oldModeColor = "GREEN";
                                                break;
                                        case TagUserData.TagMode.UnLocated:
                                                oldModeColor = "PINK";
                                                break;
                                        case TagUserData.TagMode.OutOfReach:
                                                oldModeColor = "RED";
                                                break;
                                        case TagUserData.TagMode.Inactive:
                                                oldModeColor = "GRAY";
                                                break;
```

```
                                        }
                                        switch (newMode)
                                        {
                                                case TagUserData.TagMode.Active:
                                                        newModeColor = "BLUE";
                                                        break;
                                                case
 TagUserData.TagMode.TransitingInMotionSleep:
                                                        newModeColor = "ORANGE";
                                                        break;
                                                case TagUserData.TagMode.InMotionSleep:
                                                        newModeColor = "BROWN";
                                                        break;
                                                case
 TagUserData.TagMode.TransitingNoMotionSleep:
                                                        newModeColor = "LIME";
                                                        break;
                                                case TagUserData.TagMode.NoMotionSleep:
                                                        newModeColor = "GREEN";
                                                        break;
                                                case TagUserData.TagMode.UnLocated:
                                                        newModeColor = "PINK";
                                                        break;
                                                case TagUserData.TagMode.OutOfReach:
                                                        newModeColor = "RED";
                                                        break;
                                                case TagUserData.TagMode.Inactive:
                                                        newModeColor = "GRAY";
                                                        break;
                                        }

                                        string s = "~~~~~~~~~~~~~~~~~~~~ Tag Change Mode
: " + t.ID + " " + Enum.GetName(typeof(TagUserData.TagMode), oldMode) + "(" +
oldModeColor + ")";
                                        s += " to " +
Enum.GetName(typeof(TagUserData.TagMode), newMode) + "(" + newModeColor + ")";
                                        s += " at " + DateTime.Now.ToString("dd/MM/yy
HH:mm:ss.ff");

                                        if (t.tag is Architectures.SpecV01.TagV01)
                                        {
                                                Architectures.SpecV01.TagInfoV01 info =
(t.tag as Architectures.SpecV01.TagV01).Info as Architectures.SpecV01.TagInfoV01;
                                                if (info.last_ranging_notify_packet !=
null)

                                                        s += " Last Ranging Notify at " +
info.last_ranging_notify_packet.Time.ToString("dd/MM/yy HH:mm:ss.ff");
                                        }
                                        s += " ~~~~~~~~~~~~~~~~~~~~";
                                        rtls.RTLSystem.WriteLog(s);
                                }
                                if (t.alerts.Count > 0)
                                {
                                        bool flash = false;
                                        foreach (AlertType alert_type in t.alerts)
                                        {
                                                foreach (RTLSAlert alert in
t.cell.alerts)

                                                {
```

```
                                        if (alert.AlertType == alert_type)
                                                if (alert.enable_flash)
                                                {
                                                        flash = true;
                                                        break;
                                                }
                                        }
                                        if (flash)
                                                break;
                                }
                                if (flash)
                                        t.flash_on = !t.flash_on;
                                else
                                        t.flash_on = false;
                                refresh_tag = true;
                        }
                        else
                        {
                                if (t.flash_on)
                                {
                                        t.flash_on = false;
                                        refresh_tag = true;
                                }
                        }
                        if (refresh_tag)
                        {
                                refresh = true;
                                if (!refresh_cell_list.Contains(t.cell))
                                        refresh_cell_list.Add(t.cell);
                        }
                }
                if (refresh)
                {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        {
                                cellGroupPanel.ltTags_invalidate = true;
                                cellGroupPanel.pnlMap_invalidate = true;
                                cellGroupPanel.CheckInvalidate();
                        }
                        else
                        {
                                foreach (CellPanel panel in cellPanel_List)
                                {
                                        if
(refresh_cell_list.Contains(panel.currentCell))
                                        {
                                                panel.ltTags_invalidate = true;
                                                panel.pnlMap_invalidate = true;
                                                panel.CheckInvalidate();
                                        }
                                }
                        }
                }

                if (FrmSystemMessage.IsCreated && FrmSystemMessage.HasMessage)
                {
                        lbSystemMessage.Visible = true;
```

```
                        if (lbSystemMessage.BackColor == Color.Yellow)
                        {
                                lbSystemMessage.BackColor = Color.Blue;
                                lbSystemMessage.ForeColor = Color.Yellow;
                        }
                        else
                        {
                                lbSystemMessage.BackColor = Color.Yellow;
                                lbSystemMessage.ForeColor = Color.Blue;
                        }
                }
                else
                        lbSystemMessage.Visible = false;
        }

        private void tmr5000ms_Tick(object sender, EventArgs e)
        {
#if true
                foreach (CellUserDataV02 cud in rtls.CellList)
                {
                        if (cud.masterAnchor != null && !(cud.masterAnchor is
Architectures.SpecV01.MasterDummy) && !cud.masterAnchor.PlaybackMode &&
cud.connected)
                                if (cud.waiting_cfg_response)
                                        if
(DateTime.Now.Subtract(cud.cfg_sent_time).TotalSeconds > 5)
                                        {
                                                string message = "Master Anchor
has not respond to Master Anchor Config Command since : " +
cud.cfg_sent_time.ToString("dd/MM/yy HH:mm:ss.ff");
                                                AddSystemMessage(new
SystemMessage(rtls, cud.masterAnchor, cud.masterAnchor,
SystemMessage.MessageType.MasterAnchor_No_Response, message, DateTime.Now), true);
                                        }
                }
#endif
        }

        private void OnAddRemove(Object s, AddRemoveEventArgs e)
        {
                lock (rtls_queue)
                {
                        rtls_queue.Enqueue(e);
                }
        }

        private void OnAddRemove(AddRemoveEventArgs e)
        {
                if (e.device is Anchor)
                {
                        if (e.add && e.device is
Architectures.SpecV01.MasterAnchor)
                                DisplayCellOnPanel(rtls.GetCellUserData(e.device
as Architectures.SpecV01.MasterAnchor));
                }
                else if (e.device is Tag)
                {
                        if (!e.add)
```

```
                        {
                                Tag tag = e.device as Tag;
                                if (tag.Info is TagInfo_TOA)
                                        if ((tag.Info as
TagInfo_TOA).last_ranging_notify_packet != null)
                                        {
                                                CellUserDataV02 cud = null;
                                                Cell cell = (tag.Info as
TagInfo_TOA).last_ranging_notify_packet.FromCell;
                                                if (cell != null)
                                                {
                                                        if (cell.Controller is
Architectures.SpecV01.MasterAnchor)
                                                                cud =
rtls.GetCellUserData(cell.Controller as Architectures.SpecV01.MasterAnchor);
                                                        else
                                                                cud =
rtls.GetCellUserData(cell.ID);
                                                }
                                                if (cud != null)
                                                {
                                                        if
(rtls.Config.CellDisplayMode == SystemConfigV02.CellPanelDisplayMode.Global)
                                                        {

        cellGroupPanel.FillTagList();

        cellGroupPanel.pnlMap_invalidate = true;

        cellGroupPanel.CheckInvalidate();
                                                        }
                                                        else
                                                        {
                                                                foreach (CellPanel
panel in cellPanel_List)
                                                                {
                                                                        if
(panel.currentCell == cud)
                                                                        {

        panel.FillTagList();

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }
#if false
                                if (e.device is Anchor)
                                {
                                        rtls.OnAddRemoveAnchor(e);
                                        if (e.add && e.device is
Architectures.SpecV01.MasterAnchor)
```

```
                                    DisplayCellOnPanel(rtls.GetCellUserData(e.device
as Architectures.SpecV01.MasterAnchor));
                        }
                        else if (e.device is Tag)
                        {
                                if (!e.add)
                                {
                                        foreach (TagUserData tud in rtls.TagList)
                                        {
                                                if (tud.tag == e.device)
                                                {
                                                        foreach (CellPanel panel in
cellPanel_List)
                                                        {
                                                                if (panel.currentCell ==
tud.cell)
                                                                {
                                                                        panel.FillTagList();
                                                                }
                                                        }
                                                        break;
                                                }
                                        }
                                }
                                rtls.OnAddRemoveTag(e);
                        }
                        else if (e.device is Cell)
                                rtls.OnAddRemoveCell(e);
#endif
#if false
                        if (e.device is Anchor)
                        {
                                Anchor a = e.device as Anchor;
                                if (e.add)
                                {
                                        if (rtls.RTLSystem.Database != null)
                                        {
                                                if
(!rtls.RTLSystem.Database.SetAnchorProp(a))

        rtls.RTLSystem.Database.SetDefaultProp(a, false);
                                                if (a is
Architectures.SpecV01.MasterAnchor)
                                                {
                                                        Architectures.SpecV01.MasterAnchor
master = a as Architectures.SpecV01.MasterAnchor;
                                                        DeviceCommandPacket cmd = new
Architectures.SpecV01.Commands.GetVersion(this, master);
                                                        try
                                                        {
                                                                master.SendCommand(this,
cmd);
                                                        }
                                                        catch (Exception ex)
                                                        {

        master.RTLSystem.LogException(ex);
                                                        }
```

```
                                        CellUserDataV02 cell =
CellUserDataV02.LoadConfig(rtls, master);
                                        if (cell == null)
                                              cell = new
CellUserDataV02(rtls, master);
                                        cell.DownloadMasterAnchorConfig();
                                        cell.SaveMasterAnchorConfig();
                                        cell.Save();
                                        AddToCellList(cell);
                                  }
                                  else if (a is ISlaveDevice)
                                  {
                                        Device master = (a as
ISlaveDevice).GetMaster();
                                        if (master != null && master is
Architectures.SpecV01.MasterAnchor)
                                        {
                                              CellUserDataV02 cell =
rtls.GetCellUserData(master as Architectures.SpecV01.MasterAnchor);
                                              if (cell != null)
                                                    cell.Refresh();
                                        }
                                  }
                                  else if (a is
Nanotron.nanoLOC.AVR_DK.Anchor)
                                  {
                                        CellUserDataV02 cell =
rtls.GetCellUserData(rtls.NanoLocSystem.cell.ID);
                                        if (cell != null)
                                        {
                                              bool exists = false;
                                              foreach (AnchorProperty ap
in cell.anchors)
                                              {
                                                    if (ap.AnchorId ==
a.ID)
                                                    {
                                                          exists =
true;
                                                          break;
                                                    }
                                              }
                                              if (!exists)
                                              {
                                                    AnchorProperty ap =
new AnchorProperty();
                                                    ap.AnchorNumber =
(byte)(cell.anchors.Count + 1);
                                                    ap.AnchorId = a.ID;
                                                    ap.LogicalName =
a.Name;
                                                    ap.X =
(float)a.Prop.location.X;
                                                    ap.Y =
(float)a.Prop.location.Y;

      cell.anchors.Add(ap);
                                                    cell.Refresh();
```

```
                                                }
                                        }
                                }
                        }
                        else
                        {
                        }
                }
                if (e.device is Tag)
                {
                        Tag t = e.device as Tag;
                        if (e.add)
                        {
                                if (rtls.RTLSystem.Database != null)
                                {
                                        if
(!rtls.RTLSystem.Database.SetTagProp(t))

        rtls.RTLSystem.Database.SetDefaultProp(t, true);
                                        rtls.TagList.Add(new TagUserData(t,
TagUserData.TagMode.Active, DateTime.Now));
                                }
                        }
                        else
                        {
                                {
                                        foreach (TagUserData tud in rtls.TagList)
                                        {
                                                if (tud.tag == t)
                                                {
                                                        rtls.TagList.Remove(tud);
                                                        if (tud.cell != null)
                                                        {

        tud.cell.tags.Remove(tud);
                                                                foreach (CellPanel
panel in cellPanel_List)
                                                                {
                                                                        if
(panel.currentCell == tud.cell)

        panel.FillTagList();
                                                                }
                                                        }
                                                        break;
                                                }
                                        }
                                }
                        }
                }
                if (e.device is Cell)
                {
                        Cell c = e.device as Cell;
                        if (e.add)
                        {
                                c.config = new CellConfiguration();
                                if (rtls.RTLSystem.Database != null)
```

```
                                {
                                        if
(!rtls.RTLSystem.Database.SetCellProp(c))

        rtls.RTLSystem.Database.SetDefaultProp(c, false);
                                }
                                if (c is Nanotron.nanoLOC.AVR_DK.Cell)
                                {
                                        bool exists = false;
                                        foreach (CellUserDataV02 cud in
rtls.CellList)
                                        {
                                                if (cud.cell.ID == c.ID)
                                                {
                                                        exists = true;
                                                        break;
                                                }
                                        }
                                        if (!exists)
                                        {
                                                CellUserDataV02 cell = new
CellUserDataV02(rtls, c);
                                                AddToCellList(cell);
                                        }
                                }
                        }
                        else
                        {
                        }
                }
#endif
        }

        private void OnDeviceNotify(Object s, DeviceNotifyEventArgs e)
        {
                lock (rtls_queue)
                {
                        rtls_queue.Enqueue(e);
                }
        }

        private void OnDeviceNotify(DeviceNotifyEventArgs e)
        {
                if (e.notify == DeviceNotify.InformationUpdate)
                {
                }
                else if (e.notify == DeviceNotify.PropertyChange)
                {
                        if (e.device is Anchor)
                        {
                                if (e.device is IMasterSlaveDevice && (e.device
as IMasterSlaveDevice).IsMaster())
                                {
                                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                        {
                                                cellGroupPanel.pnlMap_invalidate =
true;
```

```
                                        cellGroupPanel.CheckInvalidate();
                                    }
                                    else
                                    {
                                        foreach (CellPanel panel in
cellPanel_List)
                                        {
                                            if (panel.currentCell !=
null && panel.currentCell.masterAnchor == e.device)
                                            {

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();
                                            }
                                        }
                                    }
                                }
                                else if (e.device is ISlaveDevice)
                                {
                                    Device d = (e.device as
ISlaveDevice).GetMaster();

                                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                    {
                                        cellGroupPanel.pnlMap_invalidate =
true;
                                        cellGroupPanel.CheckInvalidate();
                                    }
                                    else
                                    {
                                        foreach (CellPanel panel in
cellPanel_List)
                                        {
                                            if (panel.currentCell !=
null && panel.currentCell.masterAnchor == d)
                                            {

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();
                                            }
                                        }
                                    }
                                }
                            }
                            else if (e.device is Tag)
                            {
                                if (e.device is ISlaveDevice)
                                {
                                    Device d = (e.device as
ISlaveDevice).GetMaster();
                                    Tag tag = e.device as Tag;
                                    if (rtls.RTLSystem.Database != null)

        rtls.RTLSystem.Database.LoadTagProp(tag, tag.Prop);
```

```
                                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                        {
                                                cellGroupPanel.FillTagList();
                                                cellGroupPanel.pnlMap_invalidate =
true;
                                                cellGroupPanel.CheckInvalidate();
                                        }
                                        else
                                        {
                                                foreach (CellPanel panel in
cellPanel_List)
                                                {
                                                        if (panel.currentCell !=
null && panel.currentCell.masterAnchor == d)
                                                        {
                                                                panel.FillTagList();

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();
                                                        }
                                                }
                                        }
                                }
                        }
                        else if (e.device is Cell)
                        {
                        }
                }
#if false
                else if (e.notify == DeviceNotify.StateChange)
                {
                        if (e.device is Anchor)
                        {
                                AnchorStateChange(e.device as Anchor);
                        }
                        else if (e.device is Tag)
                        {
                                TagStateChange(e.device as Tag);
                        }
                }
#endif
        }

        private void OnDevicePacket(Object s, DevicePacketEventArgs e)
        {
                SolutionHelper.HandleDevicePacket_NetworkCheck(e.device,
e.packet, rtls.RTLSystem.WriteLog);
                lock (rtls_queue)
                {
                        rtls_queue.Enqueue(e);
                }
        }

        private void OnDevicePacket(DevicePacketEventArgs e)
        {
#if false
```

```
                    if (e.packet is
Architectures.SpecV01.Packets.RegisterRequestNotifyPacket)
                    {
                          DeviceRegisterRequest(e);
                    }
#endif
                    if (e.packet is RangingNotifyPacket)
                    {
#if false
                          RangingNotifyPacket rp = e.packet as
RangingNotifyPacket;

                          TagUserData t = rtls.GetTagUserData(rp.device.ID);
                          if (t != null)
                          {
                                t.lastRanging = rp.raw.Time;
                                if (t.cell != null)
                                      t.lastRangingCycle = t.cell.cycleCount;
                                bool valid = false;
                                if(rp.ranging != null)
                                      foreach (DeviceDistance d in rp.ranging)
                                      {
                                            if (d.Valid)
                                                  valid = true;
                                      }
                                if (valid)
                                {
                                      t.lastValidRanging = rp.raw.Time;
                                      if (t.cell != null)
                                            t.lastValidRangingCycle =
t.lastRangingCycle;
                                }
                          }
#endif
#if false
                          Architectures.SpecV01.Notifys.RangingNotify rn = (rp as
Architectures.SpecV01.Packets.RangingNotifyPacket).notify;
                          if (rn.InsufficientTdmaTime)
                          {
                                FrmSystemMessage.Add(this, new
SystemMessage(rtls, e.device as Anchor, rp.device,
SystemMessage.MessageType.Insufficient_TDMA_Time, "", DateTime.Now));
                          }
#endif
                    }
                    else if (e.packet is PositionNotifyPacket)
                    {
                          TagPositionArrive(e.packet as PositionNotifyPacket);
                    }
                    else if (e.packet is DistanceNotifyPacket)
                    {

                          TagDistanceArrive(e.packet as DistanceNotifyPacket);
                    }
                    else if (e.packet is
Architectures.SpecV01.Packets.GenericNotifyPacket)
                    {
```

```
                            Architectures.SpecV01.Packets.GenericNotifyPacket np =
e.packet as Architectures.SpecV01.Packets.GenericNotifyPacket;
                            if (np.notify is
Architectures.SpecV01.Notifys.InsufficientGuardTimeNotify)
                            {

       Architectures.SpecV01.Notifys.InsufficientGuardTimeNotify notify =
np.notify as Architectures.SpecV01.Notifys.InsufficientGuardTimeNotify;
                                string message = "";
                                if (notify.guard_time > 0)
                                        message =
Convert.ToString(notify.guard_time) + " ms";
                                else
                                        message = "1 ms";
                                AddSystemMessage(new SystemMessage(rtls,
e.device as Architectures.SpecV01.MasterAnchor, null,
SystemMessage.MessageType.Insufficient_Guard_Time, message, DateTime.Now), false);
                            }
                            else if (np.notify is
Architectures.SpecV01.Notifys.TagAlertNotify)
                            {
                                Architectures.SpecV01.Notifys.TagAlertNotify
notify = np.notify as Architectures.SpecV01.Notifys.TagAlertNotify;
                                if (notify.InsufficientTdmaTime > 0)
                                {
                                        string message =
notify.InsufficientTdmaTime.ToString() + " ms";
                                        AddSystemMessage(new SystemMessage(rtls,
e.device as Architectures.SpecV01.MasterAnchor,
rtls.RTLSystem.GetDevice<Device>(notify.id),



      SystemMessage.MessageType.Insufficient_TDMA_Time, message, DateTime.Now),
false);
                                }
                                //OnTagAlertNotify(notify);
                            }
                            else if (np.notify is
Architectures.SpecV01.Notifys.MasterResetNotify)
                            {
                                Architectures.SpecV01.Notifys.MasterResetNotify
notify = np.notify as Architectures.SpecV01.Notifys.MasterResetNotify;
#if false
                                int status = (notify.status & 0xf0) >> 4;
                                if (status == 4 || status == 5 || status == 6 ||
status == 7 || status == 9)
                                {
                                }
                                else
#endif
                                {
                                        string message = (notify.IsReset430 ?
"430 " : "340 ") + (notify.IsResetInProgress ? "Reset in Progress" : "Reset End");
                                        AddSystemMessage(new SystemMessage(rtls,
e.device as Architectures.SpecV01.MasterAnchor, e.device,
SystemMessage.MessageType.MasterAnchor_Resetting, message, DateTime.Now), false);
                                }
```

```
                        }
                    }
                    else if (e.packet is
Architectures.SpecV01.Packets.GenericResponsePacket)
                    {
                        Architectures.SpecV01.Packets.GenericResponsePacket rp
= e.packet as Architectures.SpecV01.Packets.GenericResponsePacket;
                        if (rp.notify is
Architectures.SpecV01.Notifys.ConfigMasterAnchorResponse)
                        {
                            if (e.device is
Architectures.SpecV01.MasterAnchor)
                            {

      Architectures.SpecV01.Notifys.ConfigMasterAnchorResponse response =
rp.notify as Architectures.SpecV01.Notifys.ConfigMasterAnchorResponse;
                                CellUserDataV02 cell =
rtls.GetCellUserData(e.device as Architectures.SpecV01.MasterAnchor);
                                if (cell != null)
                                {
                                    //cell.noOfRegisteredAnchor =
response.cfg.anchor_list.Count;
                                    foreach (CellPanel panel in
cellPanel_List)
                                    {
                                        if (panel.currentCell ==
cell)
                                        {

      //panel.ShowNoOfRegisteredAnchor(response.cfg.anchor_list.Count);
#if false
                                            int count = 0;
                                            foreach
(Architectures.SpecV01.MasterAnchorConfig.Anchor a in
rp.internal_config_anchor_list)
                                            {
                                                if (a.id !=
0)
      count++;
                                            }
      panel.ShowNoOfRegisteredAnchor(count);
#endif
                                            int no_of_anchor =
0;
                                            foreach (var a in
cell.cfg.anchor_list)
                                            {
                                                if (a.id !=
0)
      no_of_anchor++;
                                            }
      //panel.ShowNoOfRegisteredAnchor(cell.cfg.anchor_list.Count);
      panel.ShowNoOfRegisteredAnchor(no_of_anchor);
```

```
        panel.ShowCycleTime(response.cfg.CalCycleTime());
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                    else if (e.packet is
Architectures.SpecV01.Packets.TagRangingAlertPacketV01)
                                    {
                                            Architectures.SpecV01.Packets.TagRangingAlertPacketV01
alert = e.packet as Architectures.SpecV01.Packets.TagRangingAlertPacketV01;
                                            if (alert.msg ==
CSL.RTLS.Architectures.SpecV01.Packets.TagRangingAlertPacketV01.Message.Insufficie
ntTDMATime)
                                            {
                                                    if (alert.notify.InsufficientTdmaTime > 0)
                                                    {
                                                            string message =
alert.notify.InsufficientTdmaTime.ToString() + " ms";
                                                            AddSystemMessage(new SystemMessage(rtls,
e.device as Architectures.SpecV01.MasterAnchor,
rtls.RTLSystem.GetDevice<Device>(alert.notify.TagID),



        SystemMessage.MessageType.Insufficient_TDMA_Time, message, DateTime.Now),
false);
                                                    }
                                            }
                                    }
                                    else if (e.packet is ExceptionAlertPacket)
                                    {
                                            ExceptionAlertPacket eap = e.packet as
ExceptionAlertPacket;
                                            rtls.RTLSystem.LogException(eap.exception);
                                    }
                                    else if (e.packet is RoamingNotifyPacket)
                                    {
#if false
                                            RoamingNotifyPacket np = e.packet as
RoamingNotifyPacket;

                                            TagUserData tud = rtls.GetTagUserData(np.device.ID);
                                            if (tud != null)
                                            {
                                                    CellUserDataV02 oldCell = null;
                                                    if (np.from != null)
                                                            if (np.from.Controller is
Architectures.SpecV01.MasterAnchor)
                                                                    oldCell =
rtls.GetCellUserData(np.from.Controller as Architectures.SpecV01.MasterAnchor);
                                                    foreach (CellPanel panel in cellPanel_List)
                                                    {
                                                            if (oldCell != null && panel.currentCell
== oldCell)
                                                            {
```

```
                                        panel.FillTagList();
                                        panel.pnlMap_invalidate = true;
                                        panel.CheckInvalidate();
                                }
                                if (panel.currentCell == tud.cell)
                                {
                                        tud.visible = panel.drawOneTagOnly
? false : true;
                                        panel.FillTagList();
                                }
                        }
                }
#endif
        }
    }

    private void OnAssistantNotify(Object sender, DeviceEventArgs e)
    {
        lock (rtls_queue)
        {
            rtls_queue.Enqueue(e);
        }
    }

    private void AddLog(string str)
    {
        try
        {
#if true1
                if (logFile != RTLS.AppDataPath + @"\log\log_" +
DateTime.Today.ToString("yyyyMMdd") + ".txt")
                {
                        if (writer != null)
                                writer.Flush();
                        if (log != null)
                                log.Close();

                        logFile = RTLS.AppDataPath + @"\log\log_" +
DateTime.Today.ToString("yyyyMMdd") + ".txt";
                        log = File.Open(logFile, FileMode.Append,
FileAccess.Write, FileShare.Read);
                        writer = new StreamWriter(log);
                }
                if (writer != null)
                        writer.WriteLine(str);
#endif
        }
        catch (Exception ex)
        {
            MessageBox.Show(this, ex.Message, Text);
        }
    }

#if false
    private void
OnTagAlertNotify(Architectures.SpecV01.Notifys.TagAlertNotify notify)
    {
        CellUserDataV02 cell = null;
```

```
                    TagUserData tud =
rtls.GetTagUserData(notify.id.ToString("X12"));
                    if (tud != null)
                           cell = tud.cell;
                    if (cell != null)
                    {
                           if (cell.alert_list != null)
                                  foreach (RTLSAlert alert in cell.alert_list)
                                  {
                                         if (alert is InMotionAlert)
                                         {
                                                if (notify.MotionDetectorActive)
                                                {
                                                       if
(!tud.alerts.Contains(alert.AlertType))

       tud.alerts.Add(alert.AlertType);
                                                }
                                                else
                                                {
                                                       if
(tud.alerts.Contains(alert.AlertType))

       tud.alerts.Remove(alert.AlertType);
                                                }
                                         }
                                         else if (alert is NoMotionAlert)
                                         {
                                                if (!notify.MotionDetectorActive)
                                                {
                                                       if
(!tud.alerts.Contains(alert.AlertType))

       tud.alerts.Add(alert.AlertType);
                                                }
                                                else
                                                {
                                                       if
(tud.alerts.Contains(alert.AlertType))

       tud.alerts.Remove(alert.AlertType);
                                                }
                                         }
                                         else if (alert is BatteryAlert)
                                         {
                                                if (notify.GetVoltage() <= (alert
as BatteryAlert).threshold_voltage)
                                                {
                                                       if
(!tud.alerts.Contains(alert.AlertType))

       tud.alerts.Add(alert.AlertType);
                                                }
                                                else
                                                {
                                                       if
(tud.alerts.Contains(alert.AlertType))
```

```
        tud.alerts.Remove(alert.AlertType);
                                        }
                                    }
                                    else if (alert is HighTemperatureAlert)
                                    {
                                        if (notify.GetTemperature() >=
(alert as HighTemperatureAlert).threshold_temperature)
                                        {
                                            if
(!tud.alerts.Contains(alert.AlertType))

        tud.alerts.Add(alert.AlertType);
                                        }
                                        else
                                        {
                                            if
(tud.alerts.Contains(alert.AlertType))

        tud.alerts.Remove(alert.AlertType);
                                        }
                                    }
                                    else if (alert is LowTemperatureAlert)
                                    {
                                        if (notify.GetTemperature() <=
(alert as LowTemperatureAlert).threshold_temperature)
                                        {
                                            if
(!tud.alerts.Contains(alert.AlertType))

        tud.alerts.Add(alert.AlertType);
                                        }
                                        else
                                        {
                                            if
(tud.alerts.Contains(alert.AlertType))

        tud.alerts.Remove(alert.AlertType);
                                        }
                                    }
                                }
                            }
                        }
#endif

                        private void AnchorStateChange(Anchor anchor)
                        {
                            //rtls.OnAnchorStateChange(anchor);
                            if (anchor is Architectures.SpecV01.MasterAnchor)
                            {
#if false
                                if (anchor.Online)
                                {
                                    CellUserDataV02 cell =
rtls.GetCellUserData(anchor as Architectures.SpecV01.MasterAnchor);
                                    if (cell == null)
                                    {
```

```
                                        cell = CellUserDataV02.LoadConfig(rtls,
anchor as Architectures.SpecV01.MasterAnchor);
                                        if (cell == null)
                                        {
                                                cell = new CellUserDataV02(rtls,
anchor as Architectures.SpecV01.MasterAnchor);
                                                cell.DownloadMasterAnchorConfig();
                                                cell.SaveMasterAnchorConfig();
                                                cell.Save();
                                        }
                                        rtls.CellList.Add(cell);
                                        DisplayCellOnPanel(cell);
                                }
                                else
                                        cell.DownloadMasterAnchorConfig();
                        }
                        else
                        {
                                if (anchor is Architectures.SpecV01.MasterDummy)
                                {
                                        CellUserDataV02 cell =
rtls.GetCellUserData(anchor as Architectures.SpecV01.MasterAnchor);
                                        if (cell != null)
                                                RemoveFromCellList(cell);
                                }
                        }
#endif
                        if (!anchor.Online)
                        {
                                if (anchor is Architectures.SpecV01.MasterDummy)
                                        RemoveOfflineMasterDummyFromPanel(anchor
as Architectures.SpecV01.MasterDummy);
                        }
                        RefreshCellPanel_CellList();
                        try
                        {
                                if (!closing && !(anchor is
Architectures.SpecV01.MasterDummy))
                                {
                                        System.Net.Sockets.Socket socket =
((ITcpClientDevice)anchor).GetSocket();
                                        if (socket == null || !socket.Connected)
                                        {
                                                AddSystemMessage(new
SystemMessage(rtls, anchor as Architectures.SpecV01.MasterAnchor, null,
SystemMessage.MessageType.MasterAnchor_Disconnected, "", DateTime.Now), false);
                                                MasterAnchorMonitor monitor =
rtls.RTLSystem.GetDevice<MasterAnchorMonitor>(0);

      Architectures.SpecV01.MasterAnchorMonitorV01.Record record =
monitor.GetRecord(anchor as Architectures.SpecV01.MasterAnchor);
                                                if (record != null &&
record.server_ip != null)
                                                {
                                                        IPAddress[] list =
Dns.GetHostAddresses(Dns.GetHostName());
                                                        bool exists = false;
```

```
                                                        foreach (IPAddress ip in
list)
                                                        {
                                                                if (ip.ToString() ==
record.server_ip.ToString())
                                                                {
                                                                        exists =
true;
                                                                        break;
                                                                }
                                                        }
                                                        if (!exists)
                                                        {
                                                                string message =
"Master Anchor " + anchor.Name + "(" + anchor.ID + ") Network Connection
down.\n\n";
                                                                message += "Please
check Network Cable / Adapter.";
                                                                FrmMessage frm = new
FrmMessage(this, "", message);
                                                        }
                                                }
                                        }
                                }
                        }
                        catch (Exception)
                        {
                        }
                }
#if false
                if (anchor is ISlaveDevice)
                {
                        Device master = (anchor as ISlaveDevice).GetMaster();
                        if (master != null && master is
Architectures.SpecV01.MasterAnchor)
                        {
                                CellUserDataV02 cell =
rtls.GetCellUserData(master as Architectures.SpecV01.MasterAnchor);
                                if (cell != null)
                                        cell.Refresh();
                        }
                }
#endif
        }

        private void TagStateChange(Tag tag)
        {
                if (tag == null)
                        return;

                TagUserData tud = rtls.GetTagUserData(tag.ID);
                if (tud != null)
                {
                        if (tud.cell != null)
                        {
                                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                {
```

```
                                        cellGroupPanel.FillTagList();
                                        cellGroupPanel.pnlMap_invalidate = true;
                                        cellGroupPanel.CheckInvalidate();
                                }
                                else
                                {
                                        foreach (CellPanel panel in
cellPanel_List)
                                        {
                                                if (panel.currentCell == tud.cell)
                                                {
                                                        panel.FillTagList();
                                                        panel.pnlMap_invalidate =
true;
                                                        panel.CheckInvalidate();
                                                }
                                        }
                                }
                        }
                }

#if false
                TagUserData tud = null;
                bool exists = false;
                for (int i = 0; i < rtls.TagList.Count; i++)
                {
                        if (rtls.TagList[i].tag == tag)
                        {
                                tud = rtls.TagList[i];
                                exists = true;
                                break;
                        }
                }
                if (!exists)
                {
                        tud = new TagUserData(tag, TagUserData.TagMode.Active,
DateTime.Now);
                        rtls.TagList.Add(tud);
                }

                if (tag.Status == TagStatus.Active)
                {
                        CellUserDataV02 newCell = null;
                        CellUserDataV02 oldCell = null;

                        if (tag is ISlaveDevice)
                        {
                                Device master = (tag as
ISlaveDevice).GetMaster();
                                if (master is
Architectures.SpecV01.MasterAnchor)
                                        newCell = rtls.GetCellUserData(master as
Architectures.SpecV01.MasterAnchor);
                        }
                        else if (tag is Nanotron.nanoLOC.AVR_DK.Tag)
                        {
                                Cell c = tag.GetCell();
                                if (c != null)
```

```
                                        newCell = rtls.GetCellUserData(c.ID);
                                }
                                if (tud.cell != newCell && newCell != null)
                                {
                                        if (tud.cell != null)
                                        {
                                                oldCell = tud.cell;
                                                oldCell.tags.Remove(tud);
                                                if (oldCell.masterAnchor != null)

        rtls.ClearRegisteredTag(oldCell.masterAnchor, tag, false);
                                        }
                                        newCell.tags.Add(tud);
                                        tud.cell = newCell;

                                        foreach (CellPanel panel in cellPanel_List)
                                        {
                                                if (oldCell != null && panel.currentCell
== oldCell)
                                                {
                                                        panel.FillTagList();
                                                        panel.pnlMap_invalidate = true;
                                                        panel.CheckInvalidate();
                                                }
                                                if (panel.currentCell == tud.cell)
                                                {
                                                        tud.visible = panel.drawOneTagOnly
? false : true;
                                                        panel.FillTagList();
                                                }
                                        }
                                }
                        }
#endif
        }

#if false
                private void DeviceRegisterRequest(DevicePacketEventArgs e)
                {
                        Architectures.SpecV01.Packets.RegisterRequestNotifyPacket rn =
e.packet as Architectures.SpecV01.Packets.RegisterRequestNotifyPacket;
                        if (rn.device is Anchor)
                        {
                                if (rn.device.id == anchor_register_id)
                                        return;
                                lock (anchor_register_queue)
                                {
                                        bool idExists = false;
                                        foreach (RegisterRequest r in
anchor_register_queue)
                                        {
                                                if (r.notify.device.id == rn.device.id)
                                                {
                                                        idExists = true;
                                                        break;
                                                }
                                        }
                                        if (!idExists)
```

```
                                {
                                        anchor_register_queue.Enqueue(new
RegisterRequest(e.device as Architectures.SpecV01.MasterAnchor, rn));
                                }
                        }
                }
                else if (rn.device is Tag)
                {
                        rtls.RTLSystem.Debug_WriteLog("RTLSProgram : " +
e.packet.RawText);
                        rtls.RTLSystem.Debug_WriteLog("RTLSProgram : As Tag " +
rn.device.ID);

                        lock (tag_register_queue)
                        {
                                bool idExits = false;
                                foreach (var r in tag_register_queue)
                                {
                                        if (r.notify.device.id == rn.device.id)
                                        {
                                                idExits = true;
                                                break;
                                        }
                                }
                                if (!idExits)
                                {
                                        tag_register_queue.Enqueue(new
RegisterRequest(e.device as Architectures.SpecV01.MasterAnchor, rn));
                                }
                        }
                }
                else if (rn.device is Cell)
                {
                        rtls.RTLSystem.Debug_WriteLog("RTLSProgram : " +
e.packet.RawText);
                        rtls.RTLSystem.Debug_WriteLog("RTLSProgram : As Cell "
+ rn.device.ID);
                }
                else
                {
                        rtls.RTLSystem.Debug_WriteLog("RTLSProgram : " +
e.packet.RawText);
                        rtls.RTLSystem.Debug_WriteLog("RTLSProgram : As Unkown
" + rn.device.ID);
                }
        }
#endif

        private void DeviceRegisterRequest(RegisterRequest request)
        {
                if (request.notify.device is Anchor)
                {
                        if (request.notify.device.id == anchor_register_id)
                                return;
                        lock (anchor_register_queue)
                        {
                                bool idExists = false;
```

```
                                foreach (RegisterRequest r in
anchor_register_queue)
                                {
                                        if (r.notify.device ==
request.notify.device)
                                        {
                                                idExists = true;
                                                break;
                                        }
                                }
                                if (!idExists)
                                        anchor_register_queue.Enqueue(request);
                        }
                }
                else if (request.notify.device is Tag)
                {
                        lock (tag_register_queue)
                        {
                                bool idExits = false;
                                foreach (var r in tag_register_queue)
                                {
                                        if (r.notify.device.id ==
request.notify.device.id)
                                        {
                                                idExits = true;
                                                break;
                                        }
                                }
                                if (!idExits)
                                        tag_register_queue.Enqueue(request);
                        }
                }
        }

        private void AnchorRegisterRequest()
        {
                RegisterRequest ar;
                lock (anchor_register_queue)
                {
                        //ar = anchor_register_queue.Peek();
                        ar = anchor_register_queue.Dequeue();
                        anchor_register_id = ar.notify.device.id;
                }
                CellUserDataV02 cell = rtls.GetCellUserData(ar.masterAnchor);
                if (cell == null)
                {
                        cell = CellUserDataV02.LoadConfig(rtls,
ar.masterAnchor);
                        if (cell == null)
                                cell = new CellUserDataV02(rtls,
ar.masterAnchor);
                        rtls.AddCellUserData(cell);
                        //rtls.CellList.Add(cell);
                        DisplayCellOnPanel(cell);
                }
                if (ar.notify.notify.firmware != null)
                {
```

```
                        string version =
rtls.RTLSAssistant.VersionToString(ar.notify.notify.firmware);
                        if ((ar.notify.device as
IMasterSlaveDevice).IsMaster())
                        {
                                if (string.Compare(version,
RTLS.compatible_master_anchor_430_application_version) < 0)
                                {
                                        string message = "Master Anchor Firmware
Version Incompatible!!!\n\n";
                                        message += "Master Anchor ID No. : " +
ar.notify.device.ID + "\n\n";
                                        message += "Current Master Firmware
Version : " + version + "\n\n";
                                        message += "Required Master Firmware
Version : " + RTLS.compatible_master_anchor_430_application_version + "\n\n";
                                        MessageBox.Show(this, message, Text,
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                                }
                        }
                        else
                        {
                                if (string.Compare(version,
RTLS.compatible_slave_anchor_application_version) < 0)
                                {
                                        string message = "Slave Anchor Firmware
Version Incompatible!!!\n\n";
                                        message += "Slave Anchor ID No. : " +
ar.notify.device.ID + "\n\n";
                                        message += "Current Slave Firmware
Version : " + version + "\n\n";
                                        message += "Required Slave Firmware
Version : " + RTLS.compatible_slave_anchor_application_version + "\n\n";
                                        MessageBox.Show(this, message, Text,
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                                }
                        }
                }
                if (cell != null && cell.anchorType == AnchorType.Normal)
                {
                        if (ar.notify.notify.bootloader != null)
                        {
                                if ((ar.notify.device as
IMasterSlaveDevice).IsMaster())
                                {
                                        string version =
ar.notify.notify.bootloader.a.ToString("0");
                                        version += "." +
ar.notify.notify.bootloader.b.ToString("00");
                                        version += "." +
ar.notify.notify.bootloader.c.ToString("00");
                                        version += "." +
ar.notify.notify.bootloader.d.ToString("00");
                                        if (string.Compare(version,
RTLS.compatible_master_anchor_430_bootloader_versino) < 0)
                                        {
                                                string message = "Master Anchor
Bootloader Version Upgrade Required!!!\n\n";
```

```
                                                message += "Master Anchor ID No. :
" + ar.notify.device.ID + "\n\n";
                                                message += "Current Master
Bootloader Version : " + version + "\n\n";
                                                message += "Please contact agent
for bootloader upgrade details.\n\n";
                                                MessageBox.Show(this, message,
Text, MessageBoxButtons.OK, MessageBoxIcon.Warning);
                                        }
                                }
                        }
                        bool exists = false;
                        int slot = 0;
                        int power = 63;
                        foreach (AnchorProperty ap in cell.anchors)
                        {
                                if (ap.AnchorId !=
AnchorProperty.EmptyAnchorSlotId && ap.AnchorId == ar.notify.device.ID)
                                {
                                        exists = true;
                                        slot = ap.AnchorNumber;
                                        power = ap.Power;
                                        break;
                                }
                        }
                        if (exists)
                        {
                                try
                                {
                                        rtls.RTLSystem.WriteLog("RegisterRequest
Anchor in Config, auto confirm, anchor : " + ar.notify.device.ID + ", master : " +
ar.masterAnchor.ID);
                                        //FrmSystemMessage.Add(this, new
SystemMessage(rtls, ar.masterAnchor, ar.notify.device,
SystemMessage.MessageType.AnchorRegisterRequest, "Anchor in Config, auto confirm",
DateTime.Now), false);
                                        //ar.masterAnchor.SendCommand(this, new
Architectures.SpecV01.Commands.ConfirmRejectRegistration(this, ar.masterAnchor,
                                        //ar.notify.device.id,
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Confirm));

        //rtls.RTLSAssistant.SendConfirmRejectRegistration(this, ar.masterAnchor,
new Architectures.SpecV01.Commands.ConfirmRejectRegistration(this,
ar.masterAnchor,
                                        //ar.notify.device.id,
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Confirm));

        rtls.RTLSAssistant.SendAnchorRegistration(this, ar.masterAnchor,
ar.notify.device.id, slot, power, RegistrationOption.Accept);
                                }
                                catch (Exception ex)
                                {
                                        rtls.RTLSystem.LogException(ex);
                                }
                                cell.DownloadMasterAnchorConfig();
                        }
                        else
                        {
```

```
#if true1
                        if (cell.anchors.Count >=
Architectures.SpecV01.MasterAnchorConfig.AnchorMax)
                        {
                                rtls.RTLSystem.WriteLog("Anchor Registration
rejected due to cell full : " + ar.masterAnchor.ID + " " + ar.notify.device.ID);
                                try
                                {
                                        //ar.masterAnchor.SendCommand(this, new
Architectures.SpecV01.Commands.ConfirmRejectRegistration(this, ar.masterAnchor,
                                                //ar.notify.device.id,
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Reject));

       rtls.RTLSAssistant.SendConfirmRejectRegistration(this, ar.masterAnchor,
new Architectures.SpecV01.Commands.ConfirmRejectRegistration(this,
ar.masterAnchor,
                                                ar.notify.device.id,
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Reject));
                                }
                                catch (Exception ex)
                                {
                                        rtls.RTLSystem.LogException(ex);
                                }
                        }
                        else
#endif
                                {
                                        string masterId = ar.masterAnchor.ID;
                                        string slaveId = ar.notify.device.ID;
                                        FrmAnchorRegister frmAnchorRegister =
FrmAnchorRegister.CreateInstance(rtls, masterId, slaveId, ar.notify,


             new
FrmAnchorRegister.AnchorRegistrationFinished(AnchorRegistrationFinished));
                                        if (frmAnchorRegister != null)
                                        {
                                                string master =
frmAnchorRegister.CheckRegistration(slaveId);
                                                if (master != null && master !=
masterId)
                                                {
                                                        string message = "Anchor
with ID No. : " + slaveId + "\n\n";
                                                        message += "is already
registered with Master Anchor ID No. : " + master + "\n\n";
                                                        message += "Repeat
registration of Anchor with different Master may affect the system
performance.\n\n";
                                                        message += "Proceed Anchor
Registration?";
                                                        if (MessageBox.Show(this,
message, Text, MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation) ==
DialogResult.No)
                                                        {

       frmAnchorRegister.DeclineRegistration();
```

```
        frmAnchorRegister.Close();
                                                                return;
                                                            }
                                                        }

        rtls.RTLSystem.WriteLog("RegisterRequest Anchor not registered yet, show
panel, anchor : " + ar.notify.device.ID + ", master : " + ar.masterAnchor);
                                                        //FrmSystemMessage.Add(this, new
SystemMessage(rtls, ar.masterAnchor, ar.notify.device,
SystemMessage.MessageType.AnchorRegisterRequest, "Anchor not registered yet, show
panel", DateTime.Now), false);
                                                        frmAnchorRegister.Show(this);
                                                        return;
                                                    }
                                                    else
                                                    {

        rtls.RTLSystem.WriteLog("RegisterRequest Anchor not registered yet, but
panel is created already");
                                                        //FrmSystemMessage.Add(this, new
SystemMessage(rtls, ar.masterAnchor, ar.notify.device,
SystemMessage.MessageType.AnchorRegisterRequest, "Anchor not registered yet, but
panel is created already", DateTime.Now), false);
                                                    }
                                                }
                                            }
                                        }
                                        else
                                            rtls.RTLSAssistant.SendAnchorRegistration(this,
ar.masterAnchor, ar.notify.device.id, 0, 0, RegistrationOption.Reject);
                                        //lock (anchor_register_queue)
                                        //{
                                        //    anchor_register_queue.Dequeue();
                                        //}
                                        anchor_register_id = 0;
                                        anchor_register_flag = true;
                                    }

                                    private void AnchorRegistrationFinished(RegistrationOption
reg_option, string masterId, string slaveId, string slaveName,

                                                                                    float x,
float y, float height, float offset, int power, float azimuth, float elevation,
AnchorType type)
                                    {
                                        Architectures.SpecV01.MasterAnchor masterAnchor =
rtls.RTLSystem.GetDevice<Architectures.SpecV01.MasterAnchor>(Convert.ToUInt64(mast
erId, 16));
                                        if (reg_option == RegistrationOption.Accept)
                                        {
                                            CellUserDataV02 cell =
rtls.GetCellUserData(masterAnchor);
                                            if (cell == null)
                                            {
                                                rtls.RTLSystem.WriteLog("RegisterRequest Anchor
add failed, cell = null, master : " + masterId + ", device : " + slaveId);
                                                anchor_register_id = 0;
```

```
                                anchor_register_flag = true;
                                return;
                        }
                        int slot = cell.AddAnchor(masterAnchor.ID, slaveId,
slaveName, x, y, height, offset, power, azimuth, elevation);
                        if (slot == 0)
                        {
                                MessageBox.Show(this, "Add a new Anchor to Cell
failed. Number of Anchors in Cell reached the Maximum Limit.", Text);
                                anchor_register_id = 0;
                                anchor_register_flag = true;
                                return;
                        }
                        else if (slot < 0)
                        {
                                rtls.RTLSystem.WriteLog("RegisterRequest Anchor
add failed, slot = " + slot + ", master : " + masterId + ", device : " + slaveId);
                                anchor_register_id = 0;
                                anchor_register_flag = true;
                                return;
                        }
                        try
                        {
                                //masterAnchor.SendCommand(this, new
Architectures.SpecV01.Commands.ConfirmRejectRegistration(this, masterAnchor,
                                        //Convert.ToUInt64(slaveId, 16),
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Confirm));

        //rtls.RTLSAssistant.SendConfirmRejectRegistration(this, masterAnchor, new
Architectures.SpecV01.Commands.ConfirmRejectRegistration(this, masterAnchor,
                                        //Convert.ToUInt64(slaveId, 16),
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Confirm));
                                rtls.RTLSAssistant.SendAnchorRegistration(this,
masterAnchor, Convert.ToUInt64(slaveId, 16), slot, power,
RegistrationOption.Accept);
                        }
                        catch (Exception ex)
                        {
                                rtls.RTLSystem.LogException(ex);
                        }
#if false
                        foreach (Anchor a in rtls.RTLSystem.AnchorList)
                        {
                                if (a.ID == slaveId)
                                {

        a.RaiseNotifyEvent(DeviceNotify.PropertyChange);
                                        break;
                                }
                        }
#endif
                        Anchor anchor =
rtls.RTLSystem.GetDevice<Anchor>(Convert.ToUInt64(slaveId, 16));
                        if (anchor != null)
                        {
                                rtls.RTLSystem.Database.SetAnchorProp(anchor);

        rtls.RTLSAssistant.SaveDeviceVersionToDatabase(anchor);
```

```
                                    if (anchor is
Architectures.SpecV01.MasterAnchor)
                                        cell.anchorType = type;
                            }
                            cell.Refresh();
                            cell.DownloadMasterAnchorConfig();
                            if (rtls.Config.EnableGlobalCoordinate)
                                rtls.GlobalLocationCellGroupCellRefresh();
                            cell.active_setting.cell_group =
rtls.GlobalCoordinateCellGroup;
                            rtls.SettingChange(cell.active_setting,
"AnchorRegistrationFinished");
                            if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            {
                                cellGroupPanel.pnlMap_invalidate = true;
                                cellGroupPanel.CheckInvalidate();
                            }
                            else
                            {
                                foreach (CellPanel panel in cellPanel_List)
                                {
                                    if (panel.currentCell != null &&
panel.currentCell.masterAnchor == masterAnchor)
                                    {
                                        panel.pnlMap_invalidate = true;
                                        panel.CheckInvalidate();
                                    }
                                }
                            }
                            //if (rtls.Config.EnableAnchorRegisterSet)
                            {
                                bool exists = false;
                                foreach (AnchorRegisterSet ars in
rtls.AnchorRegisterSetList)
                                {
                                    if (ars.master_id == masterAnchor.ID)
                                    {
                                        exists = true;
                                        bool found = false;
                                        foreach (AnchorProperty ap in
ars.anchor_property_list)
                                        {
                                            if (ap.AnchorId == slaveId)
                                            {
                                                found = true;
                                                break;
                                            }
                                        }
                                        if (!found)
                                        {

        ars.anchor_property_list.Add(new AnchorProperty((byte)slot, slaveId,
slaveName, x, y, height, offset, power, azimuth, elevation));

        ars.anchor_property_list.Sort(AnchorProperty.CompareByAnchorNumber);
                                        }
                                        break;
```

```
                                        }
                                }
                                if (!exists)
                                {
                                        AnchorRegisterSet ars = new
AnchorRegisterSet(masterAnchor.ID);
                                        ars.anchor_property_list.Add(new
AnchorProperty((byte)slot, slaveId, slaveName, x, y, height, offset, power,
azimuth, elevation));
                                        rtls.AnchorRegisterSetList.Add(ars);
                                }
                        }
                }
                else
                {
                        try
                        {

        //rtls.RTLSAssistant.SendConfirmRejectRegistration(this, masterAnchor, new
Architectures.SpecV01.Commands.ConfirmRejectRegistration(this, masterAnchor,
                                        //Convert.ToUInt64(slaveId, 16),
Architectures.SpecV01.Commands.ConfirmRejectRegistration.ConfirmReject.Reject));
                                rtls.RTLSAssistant.SendAnchorRegistration(this,
masterAnchor, Convert.ToUInt64(slaveId, 16), 0, 0, reg_option);
                        }
                        catch (Exception ex)
                        {
                                rtls.RTLSystem.LogException(ex);
                        }
                }
                anchor_register_id = 0;
                anchor_register_flag = true;
        }

        private void
TagRegisterReaffirm(RTLSAssistant.TagRegisterReaffirmEventArgs e)
        {
                TagUserData tud =
rtls.GetTagUserData(e.response.notify.id.ToString("X12"));
                if (tud != null)
                {
                        if (tud.cell != null)
                        {
                                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                {
                                        cellGroupPanel.FillTagList();
                                        cellGroupPanel.pnlMap_invalidate = true;
                                        cellGroupPanel.CheckInvalidate();
                                }
                                else
                                {
                                        foreach (CellPanel panel in
cellPanel_List)
                                        {
                                                if (panel.currentCell == tud.cell)
                                                {
                                                        panel.FillTagList();
```

```
                                                panel.pnlMap_invalidate =
true;
                                                panel.CheckInvalidate();
                                        }
                                }
                        }
                }
        }

        private void TagDeregister(RTLSAssistant.TagDeregisterEventArgs e)
        {
                if (e.masterAnchor != null)
                {
                        CellUserDataV02 cud =
rtls.GetCellUserData(e.masterAnchor);
                        if (cud != null)
                        {
                                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                {
                                        cellGroupPanel.FillTagList();
                                        cellGroupPanel.pnlMap_invalidate = true;
                                        cellGroupPanel.CheckInvalidate();
                                }
                                else
                                {
                                        foreach (CellPanel panel in
cellPanel_List)
                                        {
                                                if (panel.currentCell == cud)
                                                {
                                                        panel.FillTagList();
                                                        panel.pnlMap_invalidate =
true;
                                                        panel.CheckInvalidate();
                                                }
                                        }
                                }
                        }
                }
        }

        public void TagPositionArrive(PositionNotifyPacket pp)
        {
#if false
            MessageBox.Show("false")
                if (pp.device is ISlaveDevice)
                {
                        Device master = (pp.device as
ISlaveDevice).GetMaster();
                        if (master != null && master is ICellMember)
                                if (pp.FromCell == (master as
ICellMember).GetCell())
                                {

      ((IRangingDevice)pp.device).SavePosition(pp);
                                        foreach (TagUserData tud in rtls.TagList)
```

```
                                        {
                                                if (tud != null && tud.tag ==
pp.device)
                                                {
                                                        if (pp.position != null)
                                                        {
        tud.timeUpdatePosition = pp.raw.Time;
                                                                if (pp.FromCell ==
tud.tag.GetCell())
                                                                {
        tud.used_distance.Clear();
                                                                        foreach
(ReferencePosition rp in pp.position.used_ref)
                                                                        {
        foreach (RangingFilter.DeviceDistance dd in rp.distance)
                                                                                {
        if (!tud.used_distance.Contains(dd))
        tud.used_distance.Add(dd);
                                                                                }
                                                                        }
                                                                }
                                                                foreach (CellPanel
panel in cellPanel_List)
                                                                {
                                                                        if
(panel.currentCell != null)
                                                                                if
(panel.currentCell.masterAnchor == master)
                                                                                {
        //panel.TagPositionArrive(pp);
        panel.ltTags_invalidate = true;
        panel.pnlMap_invalidate = true;
        panel.CheckInvalidate();
                                                                                }
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
#endif
                        if (pp.device is Tag)
                        {
                                if (pp.device is ICellMember)
                                        if (pp.FromCell == (pp.device as
ICellMember).GetCell())
                                        {
```

```
                                        if(pp.source is TrilaterationCC &&
pp.source.id == RTLS.LocationEngineId_TOA)

       ((IPositioningDevice)pp.device).SavePosition(pp);
                                        TagUserData tud =
rtls.GetTagUserData(pp.device.ID);

                        //THIS IS THE CODE YOU WANT TO MODIFY START
                        CellUserDataV02 cell = null;
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            cell = UserCellSelection();
                        else
                        {
                            if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                        }

                        if
(cell.cellcfg.regional_correction_table.region_list.Count == 0)
                        {

                          /*
                            if (DBExist(pp.device.ID))
                            {
                                DBUpdate(pp.device.ID, "");
                                DBAdd("taglogs", pp.device.ID, "");
                            }
                            else
                            {
                                DBAdd("taginfo", pp.device.ID, "");
                                DBAdd("taglogs", pp.device.ID, "");
                            }
                           */
                        }

                        //check to generate layout
                        if (DBCheck("3"))
                        {
                            GenerateNewLayout();
                            DBUpdateTrigger("2",
(cell.cellcfg.regional_correction_table.region_list.Count-2).ToString());
                        }

                        //check to put the space is free
                        /*if (DBCheckEmptySpace())
                        {
                            DBUpdateTrigger("2", "1");
                        }
                        else
                        {
                            DBUpdateTrigger("2", "0");
                        }*/


                        foreach (CorrectionRegion k in
cell.cellcfg.regional_correction_table.region_list.ToArray())
```

```
                        {

                            if ((Convert.ToDouble(k.box.X0) <=
Convert.ToDouble(tud.tag.GetPosition().X) && Convert.ToDouble(k.box.X1) >=
Convert.ToDouble(tud.tag.GetPosition().X)) && (Convert.ToDouble(k.box.Y0) <=
Convert.ToDouble(tud.tag.GetPosition().Y) && Convert.ToDouble(k.box.Y1) >=
Convert.ToDouble(tud.tag.GetPosition().Y)) && (k.Name.ToString() =="nf" ||
k.Name.ToString() =="Site" || k.Name.ToString() == "ConstructionZone") )
                            {
                                if (DBExist(pp.device.ID))
                                {
                                    DBUpdate(pp.device.ID, "nf");
                                }
                                else
                                {
                                    DBAdd("taginfo", pp.device.ID, "nf");

                                }
                            }
                            else if ((Convert.ToDouble(k.box.X0) <=
Convert.ToDouble(tud.tag.GetPosition().X) && Convert.ToDouble(k.box.X1) >=
Convert.ToDouble(tud.tag.GetPosition().X)) && (Convert.ToDouble(k.box.Y0) <=
Convert.ToDouble(tud.tag.GetPosition().Y) && Convert.ToDouble(k.box.Y1) >=
Convert.ToDouble(tud.tag.GetPosition().Y)) && (k.Name.ToString() != "nf" &&
k.Name.ToString() != "Site" && k.Name.ToString() != "ConstructionZone"))
                            {
                                if (DBExist(pp.device.ID))
                                {
                                    DBUpdate(pp.device.ID, k.Name.ToString());
                                    DBUpdateDecisionTable(DBCheckPhase(),
k.Name.ToString(), "0");
                                }
                                else
                                {
                                    DBAdd("taginfo", pp.device.ID,
k.Name.ToString());
                                    DBUpdateDecisionTable(DBCheckPhase(),
k.Name.ToString(), "0");
                                }
                            }
                            else
                            {
                                DBUpdateDecisionTable(DBCheckPhase(),
k.Name.ToString(), "1");
                            }
                        }

                                            if(tud != null)
//                                          foreach (TagUserData tud in rtls.TagList)
                                            {
//                                                  if (tud != null && tud.tag ==
pp.device)
                                                  {
#if NanoLoc
                                                        if (pp.source is
Nanotron.nanoLOC.AVR_DK.LocationEngine)
                                                        {
```

```
        tud.last_nanoloc_position_packet = pp;
                                                                if (pp.position !=
null)
        tud.nanoloc_position = pp.position;
                                                            }
#endif
                                                            if (pp.position != null)
                                                            {
                                                                if (pp.source is
TrilaterationCC && pp.source.id == RTLS.LocationEngineId_TOA)
                                                                {
        tud.lastPosition = pp.raw.Time;
                                                                    if (tud.cell
!= null)
        tud.lastPositionCycle = tud.cell.cycleCount;
                                                                    if
(pp.FromCell == tud.tag.GetCell())
                                                                    {
        tud.used_distance.Clear();
                                                                        if
(pp.position.used_ref != null)
        foreach (ReferencePosition_Distance rp in pp.position.used_ref)
        {
        foreach (RangingFilter.DeviceDistance dd in rp.distance)
        {
                if (!tud.used_distance.Contains(dd))
                        tud.used_distance.Add(dd);
        }
        }
                                                                    }
                                                                }
                                                                if
(rtls.Config.CellDisplayMode == SystemConfigV02.CellPanelDisplayMode.Global)
                                                                {
        cellGroupPanel.ltTags_invalidate = true;
        cellGroupPanel.pnlMap_invalidate = true;
        cellGroupPanel.CheckInvalidate();
                                                                }
                                                                else
                                                                {
```

```
                                                                    foreach
(CellPanel panel in cellPanel_List)
                                                                    {
                                                                        if
(panel.currentCell != null && panel.currentCell.cell != null &&
panel.currentCell.cell == tud.tag.GetCell())
                                                                        {

        //panel.TagPositionArrive(pp);

        panel.ltTags_invalidate = true;

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();
                                                                        }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }

        public bool DBAdd(string table, string TagID, string Location)
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;

            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                DateTime dt = DateTime.Now;

                //NEW COMMAND
                cmd.CommandText = "INSERT INTO "+table+"(ID, TagID, Location,
DateTime) VALUES(NULL, @TagID, @Location, @DateTime)";
                cmd.Prepare();
                cmd.Parameters.AddWithValue("@TagID", TagID);
                cmd.Parameters.AddWithValue("@Location", Location);

                //get date and time
                DateTime dt1 = DateTime.Now;
                string dtActual1 = dt1.ToString("yyyy-MM-dd HH:mm:ss");
                cmd.Parameters.AddWithValue("@DateTime", dtActual1);
                cmd.ExecuteNonQuery();
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;
```

```
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }

            }
            return true;

        }

        public bool DBExist(string TagID)
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from taginfo WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                // dr.Read();
                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {
                    return true;
                }
                else
                {
                    return false;
                }
             }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return true;
        }
```

```
        public bool DBUpdate(string TagID, string Location)
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";

            MySqlConnection conn = null;
            MySqlConnection conn2 = null;

            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;

                cmd.CommandText = "SELECT * from taginfo WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;

                cmd.Connection = conn;

                MySqlDataReader dr;
                dr = cmd.ExecuteReader();

                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {
                  try
                  {
                      conn2 = new MySqlConnection(cs);
                      conn2.Open();

                      MySqlCommand cmd2 = new MySqlCommand();
                      cmd2.Connection = conn2;
                      DateTime dt = DateTime.Now;
                      cmd2.CommandText = "UPDATE `taginfo` SET `DateTime`='" +
dt.ToString("yyyy-MM-dd HH:mm:ss") + "', `Location` = '"+ Location +"' WHERE
`ID`='" + dr[0] + "'";
                      cmd2.Prepare();
                      cmd2.ExecuteNonQuery();
                      return true;
                  }
                  catch (MySqlException ex)
                  {
                      MessageBox.Show("Error:" + ex.ToString());
                      return false;

                  }
                  finally
                  {
                      if (conn2 != null)
                      {
                          conn2.Close();
                      }
                  }
                }
```

```
            }
            catch (MySqlException ex)
              {
                  MessageBox.Show("Error:" + ex.ToString());
                return false;

              }
              finally
              {
                  if (conn != null)
                  {
                      conn.Close();
                  }
              }
            return true;

        }

        public bool DBCheck(string ID)
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from triggertable WHERE ID='" + ID +
"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                dr.Read();
                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {

                    if (dr[1].ToString() == "1")
                    {
                        return true;
                    }
                    else
                    {

                        return false;
                    }
                }

            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
```

```
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return true;
        }

        public string DBCheckPhase()
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from triggertable WHERE ID='5'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                //textbox contain the id or name which you want to check in table
                while (dr.Read())
                {
                    return dr[1].ToString();
                }

            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return "";
        }

        public bool GenerateNewLayout()
        {

            cell_list_source = new CellListDataSource(rtls);
```

```
            BindingSource bsCellList = new BindingSource(cell_list_source,
cell_list_source.DataMember);


            CellUserDataV02 cell = bsCellList.Current as CellUserDataV02;
            if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                cell = UserCellSelection();
            else
            {
                if (activeCellPanelIndex < cellPanel_List.Count)
                    cell = cellPanel_List[activeCellPanelIndex].currentCell;
            }

            int i = 1;
            foreach (CorrectionRegion rc in
cell.cellcfg.regional_correction_table.region_list)
            {
                //MessageBox.Show("Name:" + rc.Name);

                string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
                MySqlConnection conn = null;
                try
                {
                    conn = new MySqlConnection(cs);
                    conn.Open();

                    MySqlCommand cmd = new MySqlCommand();
                    cmd.Connection = conn;
                    cmd.CommandText = "SELECT * from layout WHERE ID ='"+(i -
2).ToString()+"'";
                    cmd.CommandType = CommandType.Text;
                    cmd.Connection = conn;
                    MySqlDataReader dr;
                    dr = cmd.ExecuteReader();
                    // dr.Read();
                    //textbox contain the id or name which you want to check in
table

                    if (rc.Name.ToString() != "Site")
                    {
                        if (rc.Name.ToString() != "ConstructionZone")
                        {
                            if (dr.Read())
                            {
                                if (dr[1].ToString() == "0")
                                {
                                    rc.Name = "";
                                }
                                else
                                {
                                    rc.Name = dr[1].ToString();
                                }

                            }
                            else
                            {
```

```
                            rc.Name = "";
                        }
                    }
                }

            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            i++;
        }
        DBUpdateTrigger("3", "0");

        return true;

    }

    public bool DBUpdateTrigger(string ID, string value)
    {
        string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
        MySqlConnection conn = null;
        MySqlCommand cmd = new MySqlCommand();

        try
        {
            conn = new MySqlConnection(cs);
            conn.Open();

            cmd.Connection = conn;

            cmd.CommandText = "UPDATE `triggertable` SET `Value`='" + value + "'
WHERE `ID`='" + ID + "'";
            cmd.CommandType = CommandType.Text;

            cmd.Connection = conn;

            cmd.ExecuteNonQuery();

        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());
        }
        finally
        {
            if (conn != null)
            {
                conn.Close();
```

```
            }
        }
        return true;
    }

    public bool DBCheckEmptySpace()
    {
        string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
        MySqlConnection conn = null;
        try
        {
            conn = new MySqlConnection(cs);
            conn.Open();

            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = conn;


            cmd.CommandText = "SELECT * from taginfo";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = conn;
            MySqlDataReader dr;
            dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                if (dr[3].ToString() != "")
                {
                    return false;
                }
            }
            return true;

        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());

        }
        finally
        {
            if (conn != null)
            {
                conn.Close();
            }
        }
        return true;
    }

    public bool DBUpdateDecisionTable(string phase, string facility, string
value)
    {
        string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
        MySqlConnection conn = null;
        try
        {
            conn = new MySqlConnection(cs);
```

```
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                if (facility != "Site" && facility != "ConstructionZone" && facility
!= "")
                {
                    cmd.CommandText = "UPDATE decision SET `Empty`='" + value + "'
WHERE Phase ='" + phase + "' AND Facility ='" + facility + "' ";
                    cmd.CommandType = CommandType.Text;
                    cmd.ExecuteNonQuery();
                }
                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }

        }


            private void TagDistanceArrive(DistanceNotifyPacket dp)
            {
                  if (dp.distance != null)
                        if (dp.device is Tag)
                              if (dp.device is ICellMember)
                              {
                                    Cell c = (dp.device as
ICellMember).GetCell();

                                    if (c != null && dp.FromCell == c)
                                    {
                                          TagUserData tud =
rtls.GetTagUserData(dp.device.ID);

                                          if (tud != null)
                                                if (tud.cell != null &&
tud.cell == rtls.GetCellUserData(c.ID))

                                                      if (dp.distance !=
null)

                                                      {

      tud.all_distance = new List<RangingFilter.DeviceDistance>(dp.distance);
                                                            foreach
(RangingFilter.DeviceDistance d in dp.distance)

                                                            {
                                                                  if
(d.device == tud.cell.masterAnchor)

                                                                  {

      tud.oneDRange = RTLSTools.ValidateRange(d.distance_xy) / 1000f;
```

```
        break;

SystemConfigV02.CellPanelDisplayMode.Global)

        cellGroupPanel.ltTags_invalidate = true;

        cellGroupPanel.pnlMap_invalidate = true;

        cellGroupPanel.CheckInvalidate();

cellPanel_List)

(panel.currentCell != null)

(panel.currentCell.cell == c)

        //panel.TagDistanceArrive(dp);

        panel.ltTags_invalidate = true;

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();

#if false

ISlaveDevice).GetMaster();

ICellMember)

ICellMember).GetCell())

rtls.TagList)

dp.device)

!= null && tud.cell.masterAnchor == master)

(dp.FromCell == tud.tag.GetCell())
                                                    }
                                                }
                                            }
                                        if (rtls.Config.CellDisplayMode ==
                                        {

                                        }
                                        else
                                        {
                                            foreach (CellPanel panel in

                                            {
                                                if

                                                    if

                                                    {

                                                    }
                                            }
                                        }
                                    }
                                }
                                if (dp.device is ISlaveDevice)
                                {
                                    Device master = (dp.device as

                                    if (master != null && master is

                                        if (dp.FromCell == (master as

                                        {
                                            foreach (TagUserData tud in

                                            {
                                                if (tud.tag ==

                                                    if (tud.cell

                                                    {
                                                        if
```

```
        tud.all_distance = new List<RangingFilter.DeviceDistance>(dp.distance);
        foreach (RangingFilter.DeviceDistance d in dp.distance)
                                                                        {
        if (d.device == master)
        {
        tud.oneDRange = MyClass.ValidateRange(d.distance_xy) / 1000f;
        break;
        }
                                                                        }
                                                                }
                                                        }
                                                        foreach (CellPanel panel in
cellPanel_List)
                                                        {
                                                                if
(panel.currentCell != null)
                                                                if
(panel.currentCell.masterAnchor == master)
                                                                {
        //panel.TagDistanceArrive(dp);
        panel.ltTags_invalidate = true;
        panel.pnlMap_invalidate = true;
        panel.CheckInvalidate();
                                                                }
                                                        }
                                                }
                                        }
#endif
                }

                private void
OnDeviceOldFirmwareVersion(RTLSAssistant.DeviceOldFirmwareVersionEventArgs e)
                {
                        FirmwareRelease master_release =
rtls.RTLSAssistant.GetDeviceVersionRelease(e.master);
                        FirmwareRelease device_relase =
rtls.RTLSAssistant.GetDeviceVersionRelease(e.old_device);
                        string message = "Master Anchor Firmware Version Release : " +
master_release.ToString() + " Device's Old Firmware Version Release : " +
device_relase.ToString();
                        AddSystemMessage(new SystemMessage(rtls, e.master,
e.old_device, SystemMessage.MessageType.Old_FirmwareVersion, message,
DateTime.Now), false);
                }

                private void DeviceRoaming(RTLSAssistant.DeviceRoamingEventArgs e)
```

```
            {
                if (e.device is Tag)
                {
                    TagUserData tud = rtls.GetTagUserData(e.device.ID);
                    if (tud != null)
                    {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        {
                            cellGroupPanel.FillTagList();
                            cellGroupPanel.pnlMap_invalidate = true;
                            cellGroupPanel.CheckInvalidate();
                        }
                        else
                        {
                            foreach (CellPanel panel in
cellPanel_List)
                            {
                                if (e.from != null &&
panel.currentCell == e.from)
                                {
                                    panel.FillTagList();
                                    panel.pnlMap_invalidate =
true;
                                    panel.CheckInvalidate();
                                }
                                if (panel.currentCell == tud.cell)
                                {
                                    tud.visible =
panel.drawOneTagOnly ? false : true;
                                    panel.FillTagList();
                                }
                            }
                        }
                    }
                }
            }

            private void AddSystemMessage(SystemMessage message, bool
critical_message)
            {
                FrmSystemMessage.Add(this, message, critical_message);
            }

            private void FrameDataRecord(RTLSAssistant.FrameDataRecordEventArgs
e)
            {
                if (e.start)
                {
                    rbtnRecordFrameDataOn.ForeColor = Color.Red;
                    Font font = rbtnRecordFrameDataOn.Font;
                    font = new Font(font.FontFamily, font.Size,
FontStyle.Bold);
                    rbtnRecordFrameDataOn.Font = font;
                    rbtnRecordFrameDataOn.Checked = true;
                    rbtnRecordFrameDataOff.Checked = false;
                }
                else
```

```
                {
                        rbtnRecordFrameDataOn.ForeColor = Color.Black;
                        Font font = rbtnRecordFrameDataOff.Font;
                        font = new Font(font.FontFamily, font.Size,
FontStyle.Regular);
                        rbtnRecordFrameDataOff.Font = font;
                        rbtnRecordFrameDataOn.Checked = false;
                        rbtnRecordFrameDataOff.Checked = true;
                }
        }


#if DEBUG
        protected override bool ProcessDialogKey(Keys keyData)
        {
                if (keyData == Keys.F11)
                {
                        try
                        {
                                rtls.FlushLog();
                                if (rtls.logFile != null && rtls.logFile != "")

        System.Diagnostics.Process.Start(rtls.logFile);
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show(this, ex.Message, Text);
                        }
                }
                if (keyData == Keys.F12)
                {
                        CSL.RTLS.Forms.FrmDebugger frm = new
CSL.RTLS.Forms.FrmDebugger(rtls.RTLSystem);
                        frm.Show(this);
                        return true;
                }
#if true1
                if (keyData == Keys.F10)
                {
                        SmartChain_WebService.FrmSmartChainWebService frm = new
SmartChain_WebService.FrmSmartChainWebService(rtls.RTLSystem);
                        frm.Show(this);
                }
#endif
                if (keyData == Keys.F9)
                {
                        try
                        {
                                using (OpenFileDialog ofd = new
OpenFileDialog())
                                {
                                        ofd.CheckFileExists = true;
                                        ofd.InitialDirectory =
Application.StartupPath;
                                        ofd.Filter = "Component Files
(*.dll)|*.dll|All Files (*.*)|*.*";
                                        ofd.Title = "Load Component File";
                                        if (ofd.ShowDialog() == DialogResult.OK)
```

```
                                        {
                                                Assembly assembly =
Assembly.LoadFile(ofd.FileName);
                                                Type type =
assembly.GetType("RTLS_Production_Software.FrmProductionSoftware");
                                                if (type != null)

        Activator.CreateInstance(type, this, rtls.RTLSystem, RTLS.AppDataPath +
@"\log");
                                        }
                                }
                        }
                        catch (Exception ex)
                        {
                                rtls.RTLSystem.LogException(ex);
                                CSL.RTLS.Forms.FrmException.ShowDialog(this,
ex);
                        }
                }

                return base.ProcessDialogKey(keyData);
        }
#endif

        private void menuItemFileImport_Click(object sender, EventArgs e)
        {
                using (FrmImportConfig frm = new FrmImportConfig(rtls))
                {
                        frm.ShowDialog(this);
                        if (frm.finished)
                                SetCellPanels();
                }
        }

        private void menuItemFileExport_Click(object sender, EventArgs e)
        {
                using (FrmExportConfig frm = new FrmExportConfig(rtls))
                        frm.ShowDialog(this);
        }

        private void menuItemFileScenario_Click(object sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (activeCellPanelIndex < cellPanel_List.Count)
                {
                        cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                        using (FrmScenario frm = new FrmScenario(rtls, cell))
                                frm.ShowDialog(this);
                }
        }

        private void menuItemFileLoadFrameNConfig_Click(object sender,
EventArgs e)
        {
                string file;
                DebugPackV02 debugPack = null;
                using (OpenFileDialog ofd = new OpenFileDialog())
```

```
                {
                        ofd.Title = "Select Frame Data and Configuration File";
                        ofd.Filter = "Xml Files(*.xml)|*.xml";
                        ofd.Filter += "|All Files(*.*)|*.*";
                        if (ofd.ShowDialog() != DialogResult.OK)
                                return;
                        file = ofd.FileName;
                        Cursor = Cursors.WaitCursor;
                        Application.DoEvents();
                        debugPack = DebugPackV02.Load(file);
                        if (debugPack == null)
                        {
                                DebugPackV01 debugPack_V01 =
DebugPackV01.Load(file);
                                if (debugPack_V01 != null)
                                        debugPack =
DebugPackV02.CreateDebugPackV02(debugPack_V01);
                        }
                        Cursor = Cursors.Default;
                }

                if (debugPack == null)
                {
                        MessageBox.Show(this, "Invalid file or file is empty.",
Text);
                        return;
                }

                if (debugPack.configs != null)
                {
                        using (FrmImportConfig frm = new FrmImportConfig(rtls,
debugPack.configs))
                        {
                                frm.ShowDialog(this);
                                if (frm.finished)
                                        SetCellPanels();
                        }
                }
                else
                        MessageBox.Show(this, "No Configuration Information in
the file.", Text);

                if (debugPack.frameData != null)
                {
                        using (SaveFileDialog sfd = new SaveFileDialog())
                        {
                                sfd.Title = "Save Frame Data Record to file";
                                sfd.AddExtension = true;
                                sfd.CheckPathExists = true;
                                sfd.DefaultExt = "txt";
                                sfd.Filter = "Text Documents(*.txt)|*.txt";
                                sfd.Filter += "|All Files(*.*)|*.*";
                                sfd.FileName = "FrameData_" +
Path.GetFileNameWithoutExtension(file) + ".txt";
                                sfd.OverwritePrompt = true;
                                if (sfd.ShowDialog() == DialogResult.OK)
                                {
```

```
                                        File.WriteAllLines(sfd.FileName,
debugPack.frameData, Encoding.ASCII);

      Architectures.SpecV01.Forms.FrmFramePlayer frm = new
Architectures.SpecV01.Forms.FrmFramePlayer();
                                        frm.Icon = RTLS.Icon;
                                        frm.Text = RTLS.Title + frm.Text;
                                        frm.ShowFrameRateMeter = (o) =>
                                        {
                                              if (rtls.RTLSystem == null)
                                                    return;
                                              var f = new
CSL.RTLS.Architectures.SpecV01.Forms.FrmFrameRateMeter(rtls.RTLSystem);
                                              f.Show(o);
                                        };
                                        frm.Show(this);
                                        frm.OpenFile(sfd.FileName);
                                  }
                            }
                      }
                      else
                            MessageBox.Show(this, "No Frame Data Record in the
file.", Text);

               }

               private void menuItemFileSaveLogFile_Click(object sender, EventArgs
e)
               {
                      try
                      {
                            using (SaveFileDialog dialog = new SaveFileDialog())
                            {
                                  dialog.Title = "Save Log";
                                  dialog.AddExtension = true;
                                  dialog.CheckPathExists = true;
                                  dialog.Filter = "Text Files (*.txt)|*.txt|All
Files (*.*)|*.*";
                                  dialog.OverwritePrompt = true;
                                  //dialog.FileName = RTLS.AppDataPath +
@"\log\RTLS_Log_" + DateTime.Now.ToString("yyyyMMdd") + ".txt";
                                  dialog.FileName =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\" +
Path.GetFileName(rtls.logFile);
                                  if (dialog.ShowDialog() == DialogResult.OK)
                                  {
                                        rtls.FlushLog();
                                        File.Copy(rtls.logFile, dialog.FileName);
                                  }
                            }
                      }
                      catch (Exception ex)
                      {
                            MessageBox.Show(this, ex.Message, Text);
                      }
               }
```

```
        private void menuItemFileSaveCriticalLogFile_Click(object sender,
EventArgs e)
        {
            try
            {
                using (SaveFileDialog dialog = new SaveFileDialog())
                {
                    dialog.Title = "Save Critical Bug Log";
                    dialog.AddExtension = true;
                    dialog.CheckPathExists = true;
                    dialog.Filter = "Text Files (*.txt)|*.txt|All
Files (*.*)|*.*";
                    dialog.OverwritePrompt = true;
                    //dialog.FileName = RTLS.AppDataPath +
@"\log\RTLS_Log_" + DateTime.Now.ToString("yyyyMMdd") + ".txt";
                    dialog.FileName =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\" +
Path.GetFileName(rtls.critical_logFile);
                    if (dialog.ShowDialog() == DialogResult.OK)
                    {
                        rtls.FlushLog();
                        File.Copy(rtls.critical_logFile,
dialog.FileName);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message, Text);
            }
        }

        private void menuItemFileExit_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void menuItemViewAnchorsPanel_Click(object sender, EventArgs
e)
        {
            if (frmAnchors == null || !frmAnchors.Created)
            {
                frmAnchors = new FrmAnchors(this, rtls);
                frmAnchors.Show();
            }
            else if (!frmAnchors.Visible)
                frmAnchors.Show();
            else if (frmAnchors.WindowState == FormWindowState.Minimized)
                frmAnchors.WindowState = FormWindowState.Normal;
            else
                frmAnchors.Activate();
        }

        private void menuItemViewTagsPanel_Click(object sender, EventArgs e)
        {
            if (frmTags == null || !frmTags.Created)
            {
                frmTags = new FrmTags(rtls);
```

```
                        frmTags.Show();
                }
                else if (!frmTags.Visible)
                        frmTags.Show();
                else if (frmTags.WindowState == FormWindowState.Minimized)
                        frmTags.WindowState = FormWindowState.Normal;
                else
                        frmTags.Activate();
        }

        private void menuItemViewSystemMessage_Click(object sender,
EventArgs e)
        {
                FrmSystemMessage.Display(this);
        }

        private void menuItemConfiguration_Click(object sender, EventArgs e)
        {
                using (FrmConfig frm = new
FrmConfig(cellPanel_List[activeCellPanelIndex].currentCell))
                {
                        frm.ShowDialog(this);
                }
        }

        private void menuItemFilter_Click(object sender, EventArgs e)
        {
                using (FrmFilter frm = new
FrmFilter(cellPanel_List[activeCellPanelIndex].currentCell))
                {
                        frm.ShowDialog(this);
                }
        }

        private void rbtnFloorMap_Click(object sender, MouseEventArgs e)
        {
                if (sender as RadioButton == rbtnFloorMapShow)
                {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        {
                                if (rtls.CellGroupConfig.site_info == null ||
rtls.CellGroupConfig.site_info.map_file == null
                                        ||
rtls.CellGroupConfig.site_info.map_file == "")
                                        drawFloorMap =
cellGroupPanel.SelectFloorMapImageFile();
                                else
                                        drawFloorMap = true;
                        }
                        else
                        {
                                if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                                {
                                        if
(cellPanel_List[activeCellPanelIndex].currentCell.site_info == null
```

```
                                                                ||
cellPanel_List[activeCellPanelIndex].currentCell.site_info.map_file == null
                                                                ||
cellPanel_List[activeCellPanelIndex].currentCell.site_info.map_file == "")
                                                        drawFloorMap =
cellPanel_List[activeCellPanelIndex].SelectFloorMapImageFile();
                                                else
                                                        drawFloorMap = true;
                                        }
                                        else
                                                drawFloorMap = false;
                                }
                        }
                        else if (sender as RadioButton == rbtnFloorMapHide)
                                drawFloorMap = false;
                        else
                                return;

                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        {
                                if (!cellGroupPanel.SetFloorMapVisible(drawFloorMap))
                                {
                                        MessageBox.Show(this, "Load Map File Error!",
Text);
                                        return;
                                }
                        }
                        else
                        {
                                if
(!cellPanel_List[activeCellPanelIndex].SetFloorMapVisible(drawFloorMap))
                                {
                                        MessageBox.Show(this, "Load Map File Error!",
Text);
                                        return;
                                }
                        }

                        rbtnFloorMapShow.Checked = drawFloorMap;
                        rbtnFloorMapHide.Checked = !drawFloorMap;

                        pnlFloorMapScaling.Enabled = drawFloorMap;
                        Display2DMapScalingValue();
                }

                private void rbtnBoundary_Click(object sender, EventArgs e)
                {
                        if (sender as RadioButton == rbtnBoundaryShow)
                        {
                                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                        drawBoundary = true;
                                else
                                {
                                        if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                                        {
```

```
                                        if
(cellPanel_List[activeCellPanelIndex].currentCell.oneDMap)
                                        {
                                                MessageBox.Show(this, "Only valid
for 2D Display.", Text);
                                                drawBoundary = false;
                                        }
                                        else
                                                drawBoundary = true;
                                }
                                else
                                        drawBoundary = false;
                        }
                }
                else if (sender as RadioButton == rbtnBoundaryHide)
                        drawBoundary = false;
                else
                        return;

                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cellGroupPanel.SetBoundaryVisible(drawBoundary);
                else

        cellPanel_List[activeCellPanelIndex].SetBoundaryVisible(drawBoundary);
                }

                private void rbtnDiagonal_Click(object sender, EventArgs e)
                {
                        if (rtls.Config.CellDisplayMode !=
SystemConfigV02.CellPanelDisplayMode.Global)
                        {
                                if (sender as RadioButton == rbtnDiagonalShow)
                                {
                                        if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                                        {
                                                if
(cellPanel_List[activeCellPanelIndex].currentCell.oneDMap)
                                                {
                                                        MessageBox.Show(this, "Only valid
for 2D Display.", Text);
                                                        drawDiagonal = false;
                                                }
                                                else
                                                        drawDiagonal = true;
                                        }
                                        else
                                                drawDiagonal = false;
                                }
                                else if (sender as RadioButton == rbtnDiagonalHide)
                                        drawDiagonal = false;
                                else
                                        return;


        cellPanel_List[activeCellPanelIndex].SetDiagonalVisible(drawDiagonal);
                        }
```

```
            }

            private void rbtnGrid_Click(object sender, EventArgs e)
            {
                  if (sender as RadioButton == rbtnGridShow)
                  {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                              drawGrid = true;
                        else
                        {
                              if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                                    drawGrid = true;
                              else
                                    drawGrid = false;
                        }
                  }
                  else if (sender as RadioButton == rbtnGridHide)
                        drawGrid = false;
                  else
                        return;

                  if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cellGroupPanel.SetGridVisible(drawGrid);
                  else

      cellPanel_List[activeCellPanelIndex].SetGridVisible(drawGrid);
            }

            private void rbtnRegion_Click(object sender, EventArgs e)
            {
                  if (sender as RadioButton == rbtnRegionShow)
                  {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                              drawRegion = true;
                        else
                        {
                              if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                              {
                                    if
(cellPanel_List[activeCellPanelIndex].currentCell.oneDMap)
                                    {
                                          MessageBox.Show(this, "Only valid
for 2D Display.", Text);
                                          drawRegion = false;
                                    }
                                    else
                                          drawRegion = true;
                              }
                              else
                                    drawRegion = false;
                        }
                  }
                  else if (sender as RadioButton == rbtnRegionHide)
```

```
                        drawRegion = false;
                    else
                        return;

                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cellGroupPanel.SetRegionVisible(drawRegion);
                    else

        cellPanel_List[activeCellPanelIndex].SetRegionVisible(drawRegion);
            }

            private void rbtnCircle_Click(object sender, EventArgs e)
            {
                if (sender as RadioButton == rbtnCircleAll)
                {
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        drawCircle = RangingCircle.Mode.All;
                    else
                    {
                        if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                            drawCircle = RangingCircle.Mode.All;
                        else
                            drawCircle = RangingCircle.Mode.None;
                    }
                }
                else if (sender as RadioButton == rbtnCircleUsed)
                {
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        drawCircle = RangingCircle.Mode.Used;
                    else
                    {
                        if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                            drawCircle = RangingCircle.Mode.Used;
                        else
                            drawCircle = RangingCircle.Mode.None;
                    }
                }
                else if (sender as RadioButton == rbtnCircleNone)
                    drawCircle = RangingCircle.Mode.None;

                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    cellGroupPanel.SetCircleMode(drawCircle);
                else

        cellPanel_List[activeCellPanelIndex].SetCircleMode(drawCircle);
            }

            private void rbtnDisplayTag_Click(object sender, EventArgs e)
            {
                if (sender as RadioButton == rbtnDisplayTagAll)
                    drawOneTagOnly = false;
                else if (sender as RadioButton == rbtnDisplayTagOne)
```

```
                        drawOneTagOnly = true;

                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cellGroupPanel.SetDisplayTag(drawOneTagOnly);
                    else

        cellPanel_List[activeCellPanelIndex].SetDisplayTag(drawOneTagOnly);
            }

            private void rbtnTagActualLocation_Click(object sender, EventArgs e)
            {
                if (sender as RadioButton == rbtnTagActualLocationShow)
                {
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        drawTagActualLocation = true;
                    else
                    {
                        if
(cellPanel_List[activeCellPanelIndex].currentCell != null)
                        {
                            if
(cellPanel_List[activeCellPanelIndex].currentCell.oneDMap)
                            {
                                MessageBox.Show(this, "Only valid
for 2D Display.", Text);
                                drawTagActualLocation = false;
                            }
                            else
                                drawTagActualLocation = true;
                        }
                        else
                            drawTagActualLocation = false;
                    }
                }
                else if (sender as RadioButton == rbtnTagActualLocationHide)
                    drawTagActualLocation = false;
                else
                    return;

                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)

        cellGroupPanel.SetTagActualLocation(drawTagActualLocation);
                else

        cellPanel_List[activeCellPanelIndex].SetTagActualLocation(drawTagActualLoca
tion);
            }


            private void rbtnRecordFrameDataOn_CheckedChanged(object sender,
EventArgs e)
            {
#if false
                if (rbtnRecordFrameDataOn.Checked)
                {
```

```
                        if (frameLog == null)
                        {
                                rtls.RTLSystem.WriteLog("Start Record Frame");
                                try
                                {
                                        if (File.Exists(RTLS.AppDataPath +
@"\log\FrameData.txt"))
                                                File.Delete(RTLS.AppDataPath +
@"\log\FrameData.txt");
                                }
                                catch (Exception ex)
                                {
                                        rtls.RTLSystem.LogException(ex);
                                }
                                frameLog = new
Architectures.SpecV01.FrameLog(RTLS.AppDataPath + @"\log\FrameData.txt");
                                frameLog.save_notify = true;
                                frameLog.save_ranging = true;
                                rbtnRecordFrameDataOn.ForeColor = Color.Red;
                                Font font = rbtnRecordFrameDataOn.Font;
                                font = new Font(font.FontFamily, font.Size,
FontStyle.Bold);
                                rbtnRecordFrameDataOn.Font = font;

                                foreach (CellUserDataV02 cell in rtls.CellList)
                                {
                                        cell.DownloadMasterAnchorConfig();
                                }
                        }
                }
                else
                {
                        rbtnRecordFrameDataOn.ForeColor = Color.Black;
                        Font font = rbtnRecordFrameDataOff.Font;
                        font = new Font(font.FontFamily, font.Size,
FontStyle.Regular);
                        rbtnRecordFrameDataOff.Font = font;
                }
#endif
        }

        private void rbtnRecordFrameDataOff_CheckedChanged(object sender,
EventArgs e)
        {
#if false
                if (rbtnRecordFrameDataOff.Checked)
                {
                        if (frameLog != null)
                        {
                                frameLog.Flush();
                                string frameFile = frameLog.file;
                                frameLog.Dispose();
                                frameLog = null;
                                string file = "";
                                using (SaveFileDialog sfd = new
SaveFileDialog())
                                {
                                        sfd.AddExtension = true;
```

```
                                        sfd.CheckPathExists = true;
                                        sfd.DefaultExt = "xml";
                                        sfd.FileName = "DataPack_" +
DateTime.Now.ToString("yyyyMMdd_HH.mm.ss") + ".xml";
                                        sfd.Filter = "Xml Files (*.xml)|*.xml|All
Files (*.*)|*.*";
                                        sfd.OverwritePrompt = true;
                                        sfd.Title = "Save Data Pack File";
                                        if (sfd.ShowDialog() == DialogResult.OK)
                                        {
                                              try
                                              {
                                                    string[] frameData =
File.ReadAllLines(frameFile, Encoding.ASCII);
                                                    ConfigurationFilesV02
configs = new ConfigurationFilesV02();

       configs.GetAllCellConfig(rtls);
                                                    DebugPackV02 pack = new
DebugPackV02(frameData, configs);
                                                    pack.Save(sfd.FileName);
                                                    File.Delete(frameFile);
                                                    file = sfd.FileName;
                                              }
                                              catch (Exception ex)
                                              {
                                                    MessageBox.Show(this,
ex.Message, Text);
                                              }
                                        }
                                  }
                                  rtls.RTLSystem.WriteLog("Stop Record Frame : " +
file);
                                  if (MessageBox.Show(this, "Do you want to save
System Log file also?", Text, MessageBoxButtons.YesNo) == DialogResult.Yes)
                                  {
                                        try
                                        {
                                              using (SaveFileDialog dialog = new
SaveFileDialog())
                                              {
                                                    dialog.Title = "Save Log";
                                                    dialog.AddExtension = true;
                                                    dialog.CheckPathExists =
true;
                                                    dialog.Filter = "Text Files
(*.txt)|*.txt|All Files (*.*)|*.*";
                                                    dialog.OverwritePrompt =
true;
                                                    //dialog.FileName =
RTLS.AppDataPath + @"\log\RTLS_Log_" + DateTime.Now.ToString("yyyyMMdd") + ".txt";
                                                    dialog.FileName =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\" +
Path.GetFileName(rtls.logFile);
                                                    if (dialog.ShowDialog() ==
DialogResult.OK)
                                                    {
                                                          rtls.FlushLog();
```

```
        File.Copy(rtls.logFile, dialog.FileName);
                                                }
                                        }
                                }
                                catch (Exception ex)
                                {
                                        MessageBox.Show(this, ex.Message,
Text);
                                }
                        }
                }
        }
#endif
        }

        private void rbtnRecordFrameDataOn_Click(object sender, EventArgs e)
        {
                if (!rtls.RTLSAssistant.frame_data_recording)
                        rtls.RTLSAssistant.StartRecordFrameData();
        }

        private void rbtnRecordFrameDataOff_Click(object sender, EventArgs
e)
        {
                if (rtls.RTLSAssistant.frame_data_recording)
                {
                        using (SaveFileDialog sfd = new SaveFileDialog())
                        {
                                sfd.AddExtension = true;
                                sfd.CheckPathExists = true;
                                sfd.DefaultExt = "xml";
                                sfd.FileName = "DataPack_" +
DateTime.Now.ToString("yyyyMMdd_HH.mm.ss") + ".xml";
                                sfd.Filter = "Xml Files (*.xml)|*.xml|All Files
(*.*)|*.*";
                                sfd.OverwritePrompt = true;
                                sfd.Title = "Save Data Pack File";
                                if (sfd.ShowDialog() == DialogResult.OK)
                                {
                                        try
                                        {
                                                if
(rtls.RTLSAssistant.StopRecordFrameData(sfd.FileName))
                                                        if (MessageBox.Show(this,
"Do you want to save System Log file also?", Text, MessageBoxButtons.YesNo) ==
DialogResult.Yes)
                                                        {
                                                                try
                                                                {
                                                                        using
(SaveFileDialog dialog = new SaveFileDialog())
                                                                        {

        dialog.Title = "Save Log";

        dialog.AddExtension = true;
```

```
        dialog.CheckPathExists = true;

        dialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";

        dialog.OverwritePrompt = true;

        dialog.FileName =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\" +
Path.GetFileName(rtls.logFile);
                                                                        if
(dialog.ShowDialog() == DialogResult.OK)
                                                                        {

        rtls.FlushLog();

        File.Copy(rtls.logFile, dialog.FileName);
                                                                        }
                                                                    }
                                                                }
                                                                catch (Exception ex)
                                                                {

        CSL.RTLS.Forms.FrmException.ShowDialog(this, ex);
                                                                }
                                                            }
                                                        }
                                                        catch (Exception ex)
                                                        {

        CSL.RTLS.Forms.FrmException.ShowDialog(this, ex);
                                                        }
                                                    }
                                                    else
                                                    {
                                                        if (MessageBox.Show(this, "Stop Record
Frame Data without saving to file?", Text, MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button2) == DialogResult.Yes)

        rtls.RTLSAssistant.StopRecordFrameData(null);
                                                    }
                                                }
                                            }
                                        }

                                        private void menuItemToolsTagDetails_Click(object sender, EventArgs
e)
                                        {
                                            FrmTagDetails frm = new FrmTagDetails(rtls, null, null);
                                            frm.Show(this);
                                        }

                                        private void menuItemToolsTagDetailsAll_Click(object sender,
EventArgs e)
                                        {
                                            List<FrmTagDetails> frmList = new List<FrmTagDetails>();
                                            List<TagUserData> list;
```

```
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        list = rtls.TagList;
                    else
                    {
                        CellUserDataV02 cell = null;
                        if (activeCellPanelIndex < cellPanel_List.Count)
                            cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                        if (cell == null || cell.tags == null)
                            return;
                        list = cell.tags;
                    }
                    foreach (TagUserData tud in list)
                    {
                        if (tud.tag != null && tud.cell != null)
                        {
                            FrmTagDetails frmDetails = null;
                            foreach (Form frm in OwnedForms)
                            {
                                if (frm is FrmTagDetails && (frm as
FrmTagDetails).device == tud.tag)
                                {
                                    frmDetails = frm as FrmTagDetails;
                                    break;
                                }
                            }
                            if (frmDetails == null)
                                frmDetails = new FrmTagDetails(rtls,
tud.cell, tud.tag);
                            frmList.Add(frmDetails);
                        }
                    }
                    Rectangle desktop = Screen.GetWorkingArea(this);
                    Form f = new FrmTagDetails(rtls, null, null);
                    int column = desktop.Width / f.Width;
                    int row = desktop.Height / f.Height;
                    if (column <= 0)
                        column = 1;
                    if (row <= 0)
                        row = 1;
                    int k = column >= row ? column : row;
                    int ratio = 0;
                    do
                    {
                        ratio++;
                    } while (ratio * column * ratio * row < frmList.Count);
                    column *= ratio;
                    row *= ratio;

                    int width, height;
                    width = desktop.Width / column;
                    height = width * f.Height / f.Width;
                    if (height * row > desktop.Height)
                    {
                        height = desktop.Height / row;
                        width = height * f.Width / f.Height;
                    }
```

```
                int count = 0;
                for (int i = 0; i < row; i++)
                {
                        for (int j = 0; j < column && count < frmList.Count;
j++)
                        {
                                frmList[count].Size = new Size(width, height);
                                if (!frmList[count].Visible)
                                        frmList[count].Show(this);
                                frmList[count].RefreshFormSize();
                                frmList[count].Location = new Point(j * width, i
* height);
                                count++;
                        }
                }
        }

        private void menuItemToolsDataTraffic_Click(object sender, EventArgs
e)
        {
                FrmDataTraffic frm = new FrmDataTraffic(rtls);
                frm.Show(this);
        }

        private void menuItemToolsSlaveAnchors_Click(object sender,
EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmSlaveAnchors frm = new FrmSlaveAnchors(rtls, cell);
                frm.Show(this);
        }

        private void menuItemToolsAnchorNetworkOptimization_Click(object
sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                //_FrmNetworkOptimization frm = new
_FrmNetworkOptimization(rtls, cell);
```

```
                FrmNetworkOptimization frm = new FrmNetworkOptimization(rtls,
cell);
                frm.Show(this);
            }

            private void menuItemToolsAnchorTagHealthStatus_Click(object sender,
EventArgs e)
            {
                FrmDeviceHealthStatus frm = new FrmDeviceHealthStatus(rtls);
                frm.Show(this);
            }

            private void menuItemToolsAnchorCommunicationTest_Click(object
sender, EventArgs e)
            {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    cell = UserCellSelection();
                else
                {
                    if (activeCellPanelIndex < cellPanel_List.Count)
                        cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmAnchorCommunicationTest frm = new
FrmAnchorCommunicationTest(rtls, cell);
                frm.Show(this);
            }

            private void menuItemToolsAnchorTagPER_Click(object sender,
EventArgs e)
            {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    cell = UserCellSelection();
                else
                {
                    if (activeCellPanelIndex < cellPanel_List.Count)
                        cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmDevicePER frm = new FrmDevicePER(rtls, cell);
                frm.Show(this);
            }

            private void menuItemToolsMultipathCharacterization_Click(object
sender, EventArgs e)
            {

          CellUserDataV02 cell = null;

                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    cell = UserCellSelection();
                else
                {
```

```
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmMultipathCharacterization frm = new
FrmMultipathCharacterization(rtls, cell);
                frm.Show(this);
        }

        private void
menuItemToolsAnchorTagPerformanceEvaluation_Click(object sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmAnchorTagPerformance frm = new
FrmAnchorTagPerformance(rtls, cell);
                frm.Show(this);
        }

        private void
menuItemToolsAnchorConfigurationOptimization_Click(object sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmAnchorConfigOptimization frm = new
FrmAnchorConfigOptimization(rtls, cell);
                frm.Show(this);
        }

        private void menuItemToolsRangingValuesOverallSummary_Click(object
sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
```

```
                FrmRangingSummary frm = new FrmRangingSummary(rtls, cell);
                frm.Show(this);
        }

        private void
menuItemToolsLocationAccuracyOverallSummary_Click(object sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmLocationSummary frm = new FrmLocationSummary(rtls, cell);
                frm.Show(this);
        }

        private void
menuItemToolsSlaveAnchorRegistrationAnalysis_Click(object sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmSlaveAnchorRegistrationRecord frm = new
FrmSlaveAnchorRegistrationRecord(rtls, cell);
                frm.Show(this);
        }

        private void menuItemToolsTagRegisterStatistic_Click(object sender,
EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmTagRegisterStatistic frm = new
FrmTagRegisterStatistic(rtls, cell);
                frm.Show(this);
        }
```

```
        private void menuItemToolsTagRegistrationAnalysis_Click(object
sender, EventArgs e)
        {
            CellUserDataV02 cell = null;
            if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                cell = UserCellSelection();
            else
            {
                if (activeCellPanelIndex < cellPanel_List.Count)
                    cell =
cellPanel_List[activeCellPanelIndex].currentCell;
            }
            FrmTagRegistrationAnalysis frm = new
FrmTagRegistrationAnalysis(rtls, cell);
            frm.Show(this);
        }

        private void menuItemToolsTagRoamingAnalysis_Click(object sender,
EventArgs e)
        {
            FrmTagRoamingAnalysis frm = new FrmTagRoamingAnalysis(rtls);
            frm.Show(this);
        }

        private void menuItemToolsSnifferListen_Click(object sender,
EventArgs e)
        {
#if false
            CellUserDataV02 cell = null;
            if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                cell = UserCellSelection();
            else
            {
                if (activeCellPanelIndex < cellPanel_List.Count)
                    cell =
cellPanel_List[activeCellPanelIndex].currentCell;
            }
            FrmSniff frm = new FrmSniff(rtls, cell);
            frm.Show(this);
#endif
            FrmSnifferListen frm = new FrmSnifferListen(rtls);
            frm.Show(this);
        }

        private void menuItemToolsSnifferRead_Click(object sender, EventArgs
e)
        {
            FrmSnifferRead frm = new FrmSnifferRead(rtls);
            frm.Show(this);
        }

        private void menuItemToolsSniffDataAnalysis_Click(object sender,
EventArgs e)
        {
            CellUserDataV02 cell = null;
```

```
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            cell = UserCellSelection();
                    else
                    {
                            if (activeCellPanelIndex < cellPanel_List.Count)
                                    cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                    }
                    FrmSniffDataAnalysis frm = new FrmSniffDataAnalysis(rtls,
cell);
                    frm.Show(this);
            }

            private void
menuItemToolsMultiCellSynchronizationAnalysis_Click(object sender, EventArgs e)
            {
                    FrmMultiCellSynchronizationAnalysis frm = new
FrmMultiCellSynchronizationAnalysis(rtls);
                    frm.Show(this);
            }

            private void menuItemToolsRangingCountDownAnalysis_Click(object
sender, EventArgs e)
            {
                    FrmRangingCountDownAnalysis frm = new
FrmRangingCountDownAnalysis(rtls);
                    frm.Show(this);
            }

            private void
menuItemToolsTagSlotSynchronizationAnalysis_Click(object sender, EventArgs e)
            {
                    CellUserDataV02 cell = null;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            cell = UserCellSelection();
                    else
                    {
                            if (activeCellPanelIndex < cellPanel_List.Count)
                                    cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                    }
                    FrmTagSlotSynchronizationAnalysis frm = new
FrmTagSlotSynchronizationAnalysis(rtls, cell);
                    frm.Show(this);
            }

            private void menuItemToolsAutomaticDiagnostic_Click(object sender,
EventArgs e)
            {
                    CellUserDataV02 cell = null;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            cell = UserCellSelection();
                    else
                    {
                            if (activeCellPanelIndex < cellPanel_List.Count)
```

```
                                    cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                        }
                        FrmAutomaticDiagnostic frm = new FrmAutomaticDiagnostic(rtls,
cell);
                        frm.Show(this);
                }

                private void menuItemToolsAutomaticSetupOptimization_Click(object
sender, EventArgs e)
                {
                        CellUserDataV02 cell = null;
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                cell = UserCellSelection();
                        else
                        {
                                if (activeCellPanelIndex < cellPanel_List.Count)
                                        cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                        }
                        FrmAutomaticSetupOptimization frm = new
FrmAutomaticSetupOptimization(rtls, cell);
                        frm.Show(this);
                }

                private void menuItemToolsInstallationWizard_Click(object sender,
EventArgs e)
                {
                }

                private void menuItemToolsPathTracker_Click(object sender, EventArgs
e)
                {
                        CellUserDataV02 cell = null;
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                cell = UserCellSelection();
                        else
                        {

                                if (activeCellPanelIndex < cellPanel_List.Count)
                                        cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                        }
                        FrmPathTracker frm = new FrmPathTracker(rtls, cell);
                        frm.Show(this);
                }

                private void menuItemToolsPathWayPointTracker_Click(object sender,
EventArgs e)
                {
                        CellUserDataV02 cell = null;
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                cell = UserCellSelection();
                        else
                        {
```

```
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmPathWayPointTracker frm = new FrmPathWayPointTracker(rtls,
cell);
                frm.Show(this);
        }

        private void menuItemToolsTagTracker_Click(object sender, EventArgs
e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmPassiveTagTracker frm = new FrmPassiveTagTracker(rtls,
cell);
                frm.Show(this);
        }

        private void menuItemToolsTagDisplacement_Click(object sender,
EventArgs e)
        {
                FrmTagDistanceTravelled frm = new
FrmTagDistanceTravelled(rtls);
                frm.Show(this);
        }

        private void menuItemToolsFramePlayer_Click(object sender, EventArgs
e)
        {
                Architectures.SpecV01.Forms.FrmFramePlayer frm = new
Architectures.SpecV01.Forms.FrmFramePlayer();
                frm.Icon = RTLS.Icon;
                frm.Text = RTLS.Title + frm.Text;
                frm.ShowFrameRateMeter = (o) =>
                {
                        if (rtls.RTLSystem == null)
                                return;
                        var f = new
CSL.RTLS.Architectures.SpecV01.Forms.FrmFrameRateMeter(rtls.RTLSystem);
                        f.Show(o);
                };
#if false
                frm.FormClosing += (ss, ee) =>
                        {

        rtls.RTLSAssistant.frame_player_list.Remove(frm);
                        };
#endif
```

```
                    frm.Show(this);
#if false
                    rtls.RTLSAssistant.frame_player_list.Add(frm);
#endif
             }

             private void menuItemToolsBatteryLifeCalculator_Click(object sender,
EventArgs e)
             {
                    CellUserDataV02 cell = null;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                           cell = UserCellSelection();
                    else
                    {
                           if (activeCellPanelIndex < cellPanel_List.Count)
                                  cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                    }
                    Architectures.SpecV01.MasterAnchorConfig cfg;
                    if (cell != null && cell.cfg != null)
                           cfg = new
Architectures.SpecV01.MasterAnchorConfig(cell.cfg);
                    else
                           cfg = new Architectures.SpecV01.MasterAnchorConfig();
                    //_FrmBatteryLifeCalculator frm = new
_FrmBatteryLifeCalculator(cfg);
                    FrmBatteryLifeCalculator frm = new
FrmBatteryLifeCalculator(cfg);
                    frm.Show(this);
             }

             private void menuItemToolsLatencyCalculator_Click(object sender,
EventArgs e)
             {
                    CellUserDataV02 cell = null;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                           cell = UserCellSelection();
                    else
                    {
                           if (activeCellPanelIndex < cellPanel_List.Count)
                                  cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                    }
                    Architectures.SpecV01.MasterAnchorConfig cfg;
                    if (cell != null && cell.cfg != null)
                           cfg = new
Architectures.SpecV01.MasterAnchorConfig(cell.cfg);
                    else
                           cfg = new Architectures.SpecV01.MasterAnchorConfig();
                    FrmLatencyCapacityCalculator frm = new
FrmLatencyCapacityCalculator(cfg);
                    frm.Show(this);
             }

             private void menuItemToolsAccelerometerSetting_Click(object sender,
EventArgs e)
```

```
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmAccelerometerSetting frm = new
FrmAccelerometerSetting(rtls, cell);
                frm.Show(this);
        }

        private void menuItemToolsAnchorRegisterSetManagement_Click(object
sender, EventArgs e)
        {
                FrmAnchorRegisterSet frm = new FrmAnchorRegisterSet(rtls);
                frm.Show(this);
        }

        private void menuItemToolsPreferredTagSetManagement_Click(object
sender, EventArgs e)
        {
                CellUserDataV02 cell = null;
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                else
                {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                                cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                }
                FrmPreferredTagSet frm = new FrmPreferredTagSet(rtls, cell);
                frm.FormClosing += (ss, ee) =>
                {
                        if (frm.DialogResult == DialogResult.OK)
                        {
                                cell.Refresh();
                                cell.Save();
                        }
                };
                frm.Show(this);
        }

        private void menuItemToolsMultiCellManagement_Click(object sender,
EventArgs e)
        {
                FrmMultiCellManagement frm = new FrmMultiCellManagement(rtls);
                frm.Show(this);
        }

        private void menuItemDebugAcceptanceTest_Click(object sender,
EventArgs e)
        {
```

```
                    CellUserDataV02 cell = null;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                    else
                    {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                            cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                    }
                    FrmAcceptanceTest frm = new FrmAcceptanceTest(rtls, cell);
                    frm.Show(this);
            }

            private void menuItemDebugDebugOptions_Click(object sender,
EventArgs e)
            {
                    FrmDebugOptions frm = new FrmDebugOptions(rtls);
                    frm.Show(this);
            }

            private void menuItemDebugReadDeviceState_Click(object sender,
EventArgs e)
            {
                    CellUserDataV02 cell = null;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cell = UserCellSelection();
                    else
                    {
                        if (activeCellPanelIndex < cellPanel_List.Count)
                            cell =
cellPanel_List[activeCellPanelIndex].currentCell;
                    }
                    if (cell != null)
                    {
                        FrmReadDeviceState frm = new FrmReadDeviceState(rtls,
cell.masterAnchor);
                        frm.Show(this);
                    }
            }

            private void menuItemDebugTagAntennaSelection_Click(object sender,
EventArgs e)
            {
                    FrmTagAntennaSelection frm = new FrmTagAntennaSelection(rtls);
                    frm.Show(this);
            }

            private void menuItemTDOA_Click(object sender, EventArgs e)
            {
                    FrmTDOADataAnalysis frm = new FrmTDOADataAnalysis(rtls);
                    frm.Show(this);
            }

            private void menuItemWizardInstallationWizard_Click(object sender,
EventArgs e)
            {
```

```
#if false
                    if (!FrmInstallationWizard.IsCreated)
                    {
                            if (FrmAnchorRegister.isCreated)
                            {
                                    string message = "Before starting Installation
Wizard, Anchor Registration Request Panel must be closed.\n\n";
                                    message += "Do you want the system to close the
Anchor Registration Request Panel?";
                                    if (MessageBox.Show(this, message, Text,
MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation) == DialogResult.No)
                                            return;
                                    else
                                    {
                                            foreach (Form f in OwnedForms)
                                            {
                                                    if (f is FrmAnchorRegister)
                                                            f.Close();
                                            }
                                            anchor_register_id = 0;
                                            anchor_register_flag = true;
                                    }
                            }

                            FrmInstallationWizard frm =
FrmInstallationWizard.CreateInstance(rtls);
                            if(frm != null)
                                    frm.Show(this);
                    }
#endif
                    FrmPreInstallationDesign frm = new
FrmPreInstallationDesign(rtls);
                    frm.Show(this);
            }

            private CellUserDataV02 UserCellSelection()
            {
                    List<string> list = new List<string>();
                    foreach (CellUserDataV02 cud in rtls.CellList)
                    {
                            list.Add(cud.ToString());
                    }
                    if(list.Count == 0)
                            return null;

                    using (FrmSelection frm = new FrmSelection("Select Cell",
"Please select a Cell", list, ""))
                    {
                            if (frm.ShowDialog() == DialogResult.OK)
                            {
                                    foreach (CellUserDataV02 cud in rtls.CellList)
                                    {
                                            if (cud.ToString() == frm.Selection)
                                                    return cud;
                                    }
                            }
                    }
```

```
                return null;
        }

        private void CellDisplayModeChanged(object sender, EventArgs e)
        {
                if (sender == menuItemCell_Global)
                {
                        menuItemCell_1x1.Checked = false;
                        menuItemCell_1x2.Checked = false;
                        menuItemCell_1x3.Checked = false;
                        menuItemCell_1x4.Checked = false;
                        menuItemCell_2x2.Checked = false;
                        menuItemCell_Global.Checked = true;
                        rtls.Config.CellDisplayMode =
SystemConfigV02.CellPanelDisplayMode.Global;
                }
                else if (sender == menuItemCell_1x1)
                {
                        menuItemCell_1x1.Checked = true;
                        menuItemCell_1x2.Checked = false;
                        menuItemCell_1x3.Checked = false;
                        menuItemCell_1x4.Checked = false;
                        menuItemCell_2x2.Checked = false;
                        menuItemCell_Global.Checked = false;
                        rtls.Config.CellDisplayMode =
SystemConfigV02.CellPanelDisplayMode._1x1;
                }
                else if (sender == menuItemCell_1x2)
                {
                        menuItemCell_1x1.Checked = false;
                        menuItemCell_1x2.Checked = true;
                        menuItemCell_1x3.Checked = false;
                        menuItemCell_1x4.Checked = false;
                        menuItemCell_2x2.Checked = false;
                        menuItemCell_Global.Checked = false;
                        rtls.Config.CellDisplayMode =
SystemConfigV02.CellPanelDisplayMode._1x2;
                }
                else if (sender == menuItemCell_1x3)
                {
                        menuItemCell_1x1.Checked = false;
                        menuItemCell_1x2.Checked = false;
                        menuItemCell_1x3.Checked = true;
                        menuItemCell_1x4.Checked = false;
                        menuItemCell_2x2.Checked = false;
                        menuItemCell_Global.Checked = false;
                        rtls.Config.CellDisplayMode =
SystemConfigV02.CellPanelDisplayMode._1x3;
                }
                else if (sender == menuItemCell_1x4)
                {
                        menuItemCell_1x1.Checked = false;
                        menuItemCell_1x2.Checked = false;
                        menuItemCell_1x3.Checked = false;
                        menuItemCell_1x4.Checked = true;
                        menuItemCell_2x2.Checked = false;
                        rtls.Config.CellDisplayMode =
SystemConfigV02.CellPanelDisplayMode._1x4;
```

```
                }
                else if (sender == menuItemCell_2x2)
                {
                        menuItemCell_1x1.Checked = false;
                        menuItemCell_1x2.Checked = false;
                        menuItemCell_1x3.Checked = false;
                        menuItemCell_1x4.Checked = false;
                        menuItemCell_2x2.Checked = true;
                        rtls.Config.CellDisplayMode =
SystemConfigV02.CellPanelDisplayMode._2x2;
                }
                SetCellPanels();
        }

        private void FrmMain_Resize(object sender, EventArgs e)
        {
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        SetCellPanelsSize();
                else
                {
                        if (cellPanel_List.Count > 0)
                        {

      pnlMaps.ScrollControlIntoView(cellPanel_List[0]);
                                SetCellPanelsSize();
                        }
                }
        }

        private void SetCellPanels()
        {
                cellPanel_List.Clear();
                pnlMaps.Controls.Clear();
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                {
                        cellGroupPanel = new CellGroupPanel(rtls);
                        pnlMaps.Controls.Add(cellGroupPanel);
                        cellPanel_Focused(cellGroupPanel);
                }
                else
                {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x1)
                                cellPanel_List.AddRange(new CellPanel[] { new
CellPanel(rtls, 0) });
                        else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x2)
                                cellPanel_List.AddRange(new CellPanel[] { new
CellPanel(rtls, 0), new CellPanel(rtls, 1) });
                        else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x3)
                                cellPanel_List.AddRange(new CellPanel[] { new
CellPanel(rtls, 0), new CellPanel(rtls, 1), new CellPanel(rtls, 2) });
                        else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._1x4 || rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode._2x2)
```

```
                                cellPanel_List.AddRange(new CellPanel[] { new
CellPanel(rtls, 0), new CellPanel(rtls, 1), new CellPanel(rtls, 2), new
CellPanel(rtls, 3) });
                        pnlMaps.Controls.AddRange(cellPanel_List.ToArray());
                }

                SetCellPanelsSize();
                if (cellPanel_List.Count > 0)
                        activeCellPanelIndex = 0;
                for (int i = 0; i < cellPanel_List.Count; i++)
                {
                        if (i == activeCellPanelIndex)
                                cellPanel_List[i].Active(true);
                        else
                                cellPanel_List[i].Active(false);
                        if(cellPanel_List[i].currentCell != null)
                                cellPanel_List[i].currentCell.Refresh();
                        cellPanel_List[i].PanelFocused += cellPanel_Focused;
                }

                if (rtls.Config.EnableGlobalCoordinate)
                        rtls.GlobalLocationCellGroupCellRefresh();
        }

        private void SetCellPanelsSize()
        {
                if (Enum.GetName(typeof(SystemConfigV02.CellPanelDisplayMode),
rtls.Config.CellDisplayMode).StartsWith("_1x"))
                {
                        for (int i = 0; i < cellPanel_List.Count; i++)
                        {
                                cellPanel_List[i].Width = pnlMaps.Width /
cellPanel_List.Count;
                                if (cellPanel_List[i].Width *
cellPanel_List.Count > pnlMaps.Width)
                                        cellPanel_List[i].Height = pnlMaps.Height
- 17;
                                else
                                        cellPanel_List[i].Height =
pnlMaps.Height;
                                cellPanel_List[i].Location = new
Point(pnlMaps.DisplayRectangle.Left + i * cellPanel_List[i].Width, 0);
                        }
                }
                else if
(Enum.GetName(typeof(SystemConfigV02.CellPanelDisplayMode),
rtls.Config.CellDisplayMode).StartsWith("_2x"))
                {
                        {
                                for (int i = 0; i < cellPanel_List.Count; i++)
                                {
                                        cellPanel_List[i].Width = pnlMaps.Width /
2;
                                        if (cellPanel_List[i].Width * 2 >
pnlMaps.Width)
                                                cellPanel_List[i].Height =
(pnlMaps.Height - 17) / 2;
                                        else
```

```
                                                cellPanel_List[i].Height =
pnlMaps.Height / 2;
                                        }
                                }
                                {
                                        int index = 0;
                                        for (int i = 0; i < cellPanel_List.Count / 2;
i++)
                                        {
                                                for (int j = 0; j < cellPanel_List.Count
/ 2; j++)
                                                {
                                                        cellPanel_List[index].Location =
new Point(pnlMaps.DisplayRectangle.Left + j * cellPanel_List[index].Width, i *
cellPanel_List[index].Height);
                                                        index++;
                                                }
                                        }
                                }
                        }
                        else if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        {
                                cellGroupPanel.Width = pnlMaps.Width;
                                if (cellGroupPanel.Width > pnlMaps.Width)
                                        cellGroupPanel.Height = pnlMaps.Height - 17;
                                else
                                        cellGroupPanel.Height = pnlMaps.Height;
                        }
                }

                private void DisplayCellOnPanel(CellUserDataV02 cell)
                {
                        if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                cellGroupPanel.LoadCellList();
                        else
                        {
                                if (cell == null)
                                        return;

                                for (int i = 0; i < cellPanel_List.Count; i++)
                                {
                                        cellPanel_List[i].cell_list_source.Refresh();

        cellPanel_List[i].bsCellList.ResetBindings(false);
                                }
                                int idx = rtls.CellDisplayList.IndexOf(cell.ID);
                                if (idx < 0)
                                        idx = rtls.CellDisplayList.IndexOf("");
                                if (idx >= 0 && idx < cellPanel_List.Count)
                                        cellPanel_List[idx].SetSelectedCell(cell);
                        }
                }

                private void
RemoveOfflineMasterDummyFromPanel(Architectures.SpecV01.MasterDummy dummy)
                {
```

```
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cellGroupPanel.LoadCellList();
                    else
                    {
                        for (int i = 0; i < cellPanel_List.Count; i++)
                        {
                            cellPanel_List[i].cell_list_source.Refresh();

      cellPanel_List[i].bsCellList.ResetBindings(false);
                            if (cellPanel_List[i].currentCell != null &&
cellPanel_List[i].currentCell.masterAnchor == dummy)
                            {
                                rtls.CellDisplayList[i] = "";
                                cellPanel_List[i].SetSelectedCell(null);
                            }
                        }
                    }
                }

                private void RefreshCellPanel_CellList()
                {
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                        cellGroupPanel.LoadCellList();
                    else
                    {
                        for (int i = 0; i < cellPanel_List.Count; i++)
                        {
                            cellPanel_List[i].cell_list_source.Refresh();

      cellPanel_List[i].bsCellList.ResetBindings(false);
                            if (rtls.CellDisplayList[i] == "")
                                cellPanel_List[i].SetSelectedCell(null);
                        }
                    }
                }

                private void cellPanel_Focused(object sender)
                {
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    {
                        this.drawFloorMap = cellGroupPanel.drawFloorMap;
                        this.drawBoundary = cellGroupPanel.drawBoundary;
                        this.drawGrid = cellGroupPanel.drawGrid;
                        this.drawRegion = cellGroupPanel.drawRegion;
                        this.drawOneTagOnly = cellGroupPanel.drawOneTagOnly;
                        this.drawTagActualLocation =
cellGroupPanel.drawTagActualLocation;
                        this.drawCircle = cellGroupPanel.drawCircle;
                        Config2DMapButtons();
                        cellGroupPanel.Invalidate();
                    }
                    else
                    {
                        foreach (CellPanel panel in cellPanel_List)
                        {
```

```
                                if (sender == panel)
                                {
                                        activeCellPanelIndex = panel.panelIndex;
                                        panel.Active(true);
                                        this.drawFloorMap = panel.drawFloorMap;
                                        this.drawBoundary = panel.drawBoundary;
                                        this.drawGrid = panel.drawGrid;
                                        this.drawDiagonal = panel.drawDiagonal;
                                        this.drawRegion = panel.drawRegion;
                                        this.drawOneTagOnly =
panel.drawOneTagOnly;
                                        this.drawTagActualLocation =
panel.drawTagActualLocation;
                                        this.drawCircle = panel.drawCircle;
                                        Config2DMapButtons();
                                }
                                else
                                        panel.Active(false);

                                panel.Invalidate();
                        }
                }
        }

        private void Config2DMapButtons()
        {
                if (drawFloorMap)
                {
                        pnlFloorMapScaling.Enabled = true;
                        rbtnFloorMapShow.Checked = true;
                }
                else
                {
                        pnlFloorMapScaling.Enabled = false;
                        rbtnFloorMapHide.Checked = true;
                }
                Display2DMapScalingValue();
                if (drawBoundary)
                        rbtnBoundaryShow.Checked = true;
                else
                        rbtnBoundaryHide.Checked = true;
                if (drawDiagonal)
                        rbtnDiagonalShow.Checked = true;
                else
                        rbtnDiagonalHide.Checked = true;
                if (drawGrid)
                        rbtnGridShow.Checked = true;
                else
                        rbtnGridHide.Checked = true;
                if (drawRegion)
                        rbtnRegionShow.Checked = true;
                else
                        rbtnRegionHide.Checked = true;
                if (drawCircle == RangingCircle.Mode.All)
                        rbtnCircleAll.Checked = true;
                else if (drawCircle == RangingCircle.Mode.Used)
                        rbtnCircleUsed.Checked = true;
                else if (drawCircle == RangingCircle.Mode.None)
```

```
                        rbtnCircleNone.Checked = true;
                if (drawOneTagOnly)
                        rbtnDisplayTagOne.Checked = true;
                else
                        rbtnDisplayTagAll.Checked = true;
                if (drawTagActualLocation)
                        rbtnTagActualLocationShow.Checked = true;
                else
                        rbtnTagActualLocationHide.Checked = true;
        }

        private void Display2DMapScalingValue()
        {
                try
                {
                        numFloorMapOriginX.ValueChanged -=
numFloorMapOriginX_ValueChanged;
                        numFloorMapOriginY.ValueChanged -=
numFloorMapOriginY_ValueChanged;
                        numFloorMapWidth.ValueChanged -=
numFloorMapWidth_ValueChanged;
                        if (drawFloorMap)
                        {
                                float x = 0, y = 0, w = 0;
                                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                {
                                        x = cellGroupPanel.GetFloorMapOrigin().X;
                                        y = cellGroupPanel.GetFloorMapOrigin().Y;
                                        w = cellGroupPanel.GetFloorMapWidth();
                                }
                                else
                                {
                                        x =
cellPanel_List[activeCellPanelIndex].GetFloorMapOrigin().X;
                                        y =
cellPanel_List[activeCellPanelIndex].GetFloorMapOrigin().Y;
                                        w =
cellPanel_List[activeCellPanelIndex].GetFloorMapWidth();
                                }
                                numFloorMapOriginX.Value = (decimal)x;
                                numFloorMapOriginY.Value = (decimal)y;
                                if ((decimal)w >= numFloorMapWidth.Minimum)
                                        numFloorMapWidth.Value = (decimal)w;
                                else
                                        numFloorMapWidth.Value =
numFloorMapWidth.Minimum;
                        }
                        else
                        {
                                numFloorMapOriginX.Value = 0;
                                numFloorMapOriginY.Value = 0;
                                numFloorMapWidth.Value = 0.1M;
                        }
                }
                finally
                {
```

```
                        numFloorMapOriginX.ValueChanged +=
numFloorMapOriginX_ValueChanged;
                        numFloorMapOriginY.ValueChanged +=
numFloorMapOriginY_ValueChanged;
                        numFloorMapWidth.ValueChanged +=
numFloorMapWidth_ValueChanged;
                }
        }

        private void pnlLegendPaint(Graphics g)
        {
                pnlLegend.DrawBackground(g, pnlLegend.BackColor);
                DrawTagLegend(g, 6, 1, Color.White, Color.SteelBlue, "",
"Active");
                DrawTagLegend(g, 6, 18, Color.White, Color.Orange, "",
"Transiting to In Motion Sleep");
                DrawTagLegend(g, 6, 35, Color.White, Color.Brown, "", "In
Motion Sleep");
                DrawTagLegend(g, 6, 52, Color.White, Color.Lime, "",
"Transiting to No Motion Sleep");
                DrawTagLegend(g, 6, 69, Color.White, Color.Green, "", "No
Motion Sleep");
                DrawTagLegend(g, 6, 86, Color.White, Color.Orchid, "", "Cannot
Calculate Location");
                DrawTagLegend(g, 6, 103, Color.White, Color.Red, "", "Out of
Reach");
                DrawTagLegend(g, 6, 120, Color.White, Color.Gray, "", "Check
Net");
                DrawTagLegend(g, 6, 137, Color.White, Color.Black, "", "Actual
Location");
        }

        private void DrawTagLegend(Graphics g, int x, int y, Color
foreColor, Color backColor, string caption, string text)
        {
                //pnlLegend.FillCircleWithText(g, x + 10, y + 10, foreColor,
backColor, caption, 10);
                //pnlLegend.DrawText(g, x + 27, y + 3, Color.Black, text, 9f,
"Arial", StringFormat.GenericDefault);
                pnlLegend.FillCircleWithText(g, x + 10, y + 10, foreColor,
backColor, caption, 8);
                pnlLegend.DrawText(g, x + 27, y + 3, Color.Black, text, 8f,
"Arial", StringFormat.GenericDefault);

        }

        private void numFloorMapOriginX_ValueChanged(object sender,
EventArgs e)
        {
                PointF origin = new PointF((float)numFloorMapOriginX.Value,
(float)numFloorMapOriginY.Value);
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                {
                        cellGroupPanel.SetFloorMapOrigin(origin);
                        cellGroupPanel.pnlMap_invalidate = true;
                        cellGroupPanel.CheckInvalidate();
                }
```

```
                    else
                    {

        cellPanel_List[activeCellPanelIndex].SetFloorMapOrigin(origin);
                            cellPanel_List[activeCellPanelIndex].pnlMap_invalidate
= true;
                            cellPanel_List[activeCellPanelIndex].CheckInvalidate();
                    }
             }

             private void numFloorMapOriginY_ValueChanged(object sender,
EventArgs e)
             {
                    PointF origin = new PointF((float)numFloorMapOriginX.Value,
(float)numFloorMapOriginY.Value);
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    {
                            cellGroupPanel.SetFloorMapOrigin(origin);
                            cellGroupPanel.pnlMap_invalidate = true;
                            cellGroupPanel.CheckInvalidate();
                    }
                    else
                    {

        cellPanel_List[activeCellPanelIndex].SetFloorMapOrigin(origin);
                            cellPanel_List[activeCellPanelIndex].pnlMap_invalidate
= true;
                            cellPanel_List[activeCellPanelIndex].CheckInvalidate();
                    }
             }

             private void numFloorMapWidth_ValueChanged(object sender, EventArgs
e)
             {
                    float width = (float)numFloorMapWidth.Value;
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                    {
                            cellGroupPanel.SetFloorMapWidth(width);
                            cellGroupPanel.pnlMap_invalidate = true;
                            cellGroupPanel.CheckInvalidate();
                    }
                    else
                    {

        cellPanel_List[activeCellPanelIndex].SetFloorMapWidth(width);
                            cellPanel_List[activeCellPanelIndex].pnlMap_invalidate
= true;
                            cellPanel_List[activeCellPanelIndex].CheckInvalidate();
                    }
             }

             private void btnFloorMapSave_Click(object sender, EventArgs e)
             {
                    if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            cellGroupPanel.SaveFloorMapScaling();
```

```
                else

        cellPanel_List[activeCellPanelIndex].SaveFloorMapScaling();
            }

            private void btnFloorMapRestore_Click(object sender, EventArgs e)
            {
                if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                {
                    cellGroupPanel.RestoreFloorMapScaling();
                    cellGroupPanel.pnlMap_invalidate = true;
                    cellGroupPanel.CheckInvalidate();
                }
                else
                {

        cellPanel_List[activeCellPanelIndex].RestoreFloorMapScaling();
                    cellPanel_List[activeCellPanelIndex].pnlMap_invalidate
= true;
                    cellPanel_List[activeCellPanelIndex].CheckInvalidate();
                }
                Display2DMapScalingValue();
            }

            #region NanoLoc_AVR_DK

            private void menuItemNanoLocSystem_Click(object sender, EventArgs e)
            {
#if NanoLoc
                FrmNanoLocSystem frm = null;
                foreach (var f in OwnedForms)
                {
                    if (f is FrmNanoLocSystem)
                    {
                        frm = (FrmNanoLocSystem)f;
                        break;
                    }
                }
                if (frm == null)
                    frm = new FrmNanoLocSystem(rtls);
                frm.Show(this);
#endif
            }

            private void menuItemNanoLocEngineOnly_Click(object sender,
EventArgs e)
            {
                //Nanotron.nanoLOC.AVR_DK.FrmLocationEngineServer frm = new
Nanotron.nanoLOC.AVR_DK.FrmLocationEngineServer(rtls.RTLSystem,
rtls.RTLSystem.AnchorList, rtls.RTLSystem.TagList);
#if NanoLoc
                FrmNanoLocLocationEngine frm = null;
                foreach (var f in OwnedForms)
                {
                    if (f is FrmNanoLocLocationEngine)
                    {
                        frm = (FrmNanoLocLocationEngine)f;
```

```
                                    break;
                                }
                            }
                            if(frm == null)
                                frm = new FrmNanoLocLocationEngine(rtls);
                            frm.Show(this);
#endif
            }

            #endregion

        private void addObservationAreasToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            cell_list_source = new CellListDataSource(rtls);
            BindingSource bsCellList = new BindingSource(cell_list_source,
cell_list_source.DataMember);


            CellUserDataV02 cell = bsCellList.Current as CellUserDataV02;
            if (rtls.Config.CellDisplayMode ==
SystemConfigV02.CellPanelDisplayMode.Global)
                cell = UserCellSelection();
            else
            {
                if (activeCellPanelIndex < cellPanel_List.Count)
                    cell = cellPanel_List[activeCellPanelIndex].currentCell;
            }

            if (cell.cellcfg.regional_correction_table != null &&
cell.cellcfg.regional_correction_table.region_list != null)
            {
                ulong[] anchorId_List = null;
                if (cell.anchors != null)
                {
                    anchorId_List = new ulong[cell.anchors.Count];
                    for (int i = 0; i < cell.anchors.Count; i++)
                    {
                        if (cell.anchors[i].AnchorId != "")
                            anchorId_List[i] =
Convert.ToUInt64(cell.anchors[i].AnchorId, 16);
                    }
                }
                if (cell.regionalCharacterization_TagTable == null)
                    cell.regionalCharacterization_TagTable = new
List<List<string>>();
                regionalCharacterizeTable.Clear();
                for (int i = 0; i <
cell.cellcfg.regional_correction_table.region_list.Count; i++)
                {
                    Tag[] tags = null;
                    if (cell.regionalCharacterization_TagTable.Count > 0)
                    {
                        tags = new
Tag[cell.regionalCharacterization_TagTable.Count];
                        for (int n = 0; n <
cell.regionalCharacterization_TagTable.Count; n++)
```

```
                            if (i <
cell.regionalCharacterization_TagTable[n].Count)
                            {
                                ulong id;
                                if
(!ulong.TryParse(cell.regionalCharacterization_TagTable[n][i],
System.Globalization.NumberStyles.HexNumber, null, out id))
                                    tags[n] = null;
                                else
                                    tags[n] =
cell.rtls.RTLSystem.GetDevice<Tag>(id);
                            }
                    }
                    regionalCharacterizeTable.Add(new
RegionalCharacterization(rtls.RTLSystem,
cell.cellcfg.regional_correction_table.region_list[i],

tags, anchorId_List));
                }
            }



            using (FrmSubRegionDefinition frm = new FrmSubRegionDefinition(cell,
regionalCharacterizeTable))
            {
                if (frm.ShowDialog(this) == DialogResult.OK)
                    regionalCharacterizeTable = frm.regionalCharacterizeTable;
            }
        }


       }
}
```

Navisworks Manage 2013 “Schedule Checker” File:

```
//-----------------------------------------------------------------------
// NavisWorks Sample code
//-----------------------------------------------------------------------

// (C) Copyright 2009 by Autodesk Inc.

// Permission to use, copy, modify, and distribute this software in
// object code form for any purpose and without fee is hereby granted,
// provided that the above copyright notice appears in all copies and
// that both that copyright notice and the limited warranty and
// restricted rights notice below appear in all supporting
// documentation.

// AUTODESK PROVIDES THIS PROGRAM "AS IS" AND WITH ALL FAULTS.
// AUTODESK SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OF
// MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE.  AUTODESK
// DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAM WILL BE
// UNINTERRUPTED OR ERROR FREE.
//-----------------------------------------------------------------------
//
// This sample demonstrates basic LINQ style searching capabilities
// in the .NET API
//
//-----------------------------------------------------------------------

using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Data;
using System.Net;
using System.Linq;
using System.Text;
using MySql.Data;
using MySql.Data.MySqlClient;
using Autodesk.Navisworks.Api;
using Autodesk.Navisworks.Api.Plugins;
using Autodesk.Navisworks.Api.Controls;
using ComApi = Autodesk.Navisworks.Api.Interop.ComApi;
using ComApiBridge = Autodesk.Navisworks.Api.ComApi;
using Autodesk.Navisworks.Api.Interop.ComApi;
using Autodesk.Navisworks.Api.Timeliner;


namespace Examiner
{
   public partial class SearchForm : Form
   {
       bool runScript = false;
       string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
       MySqlConnection conn = null;
       MySqlCommand cmd = new MySqlCommand();
       int currentActivity = 0;
       int currentPhase = 0;
```

```
        bool overLap = false;

       public SearchForm()
       {
          InitializeComponent();
       }

       public void startend_Click(object sender, EventArgs e)
       {
           if (runScript)
           {
               runScript = false;
               MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow,
"Stopped");
               timer.Stop();
           }
           else
           {
               runScript = true;
               MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow,
"Started");
               timer.Start();

               try
               {
                   conn = new MySqlConnection(cs);
                   conn.Open();
                }
                catch (MySqlException ex)
                {
                    MessageBox.Show("Error:" + ex.ToString());

                }
           }

       }

       public void timer_Tick(object sender, EventArgs e)
       {
           if (runScript)
           {
               DBCheckDecisionTable();
               DBPhaseUpdate((currentPhase + 1).ToString());
               DocumentTimeliner doc_timeliner =
Autodesk.Navisworks.Api.Application.MainDocument.GetTimeliner();
               int countTask = doc_timeliner.Tasks.Count();
               int i = 0;

               foreach (TimelinerTask task in doc_timeliner.Tasks)
               {
                   int childCount = 1;
                   foreach (TimelinerTask child in task.Children)
                   {
                       DateTime ActualEnd = DateTime.Parse(DumpTaskInfo(child, 0,
"AE"));
                       DateTime ActualStart = DateTime.Parse(DumpTaskInfo(child, 0,
"AS"));
                       DateTime Today = DateTime.Now;
```

```
                    if (ActualStart <= Today && ActualEnd <= Today)
                    {
                        DBUpdateDecisionTable(currentPhase.ToString(),
childCount.ToString(), "1");
                    }
                    childCount++;
                }

                if (task.Children.Count == 0)
                {
                    overLap = true;
                }


                if (overLap)
                {
                    DateTime ActualEnd = DateTime.Parse(DumpTaskInfo(task, 0,
"AE"));
                    DateTime ActualStart = DateTime.Parse(DumpTaskInfo(task, 0,
"AS"));
                    DateTime Today = DateTime.Now;
                    if (ActualStart <= Today && ActualEnd <= Today)
                    {
                        DBUpdateDecisionTable((currentPhase + 1).ToString(),
"1", "1");
                        currentActivity++;
                    }

                    overLap = false;
                }

                if (currentActivity == i)
                {
                    //DATES COMPARE
                    DateTime ActualEnd = DateTime.Parse(DumpTaskInfo(task, 0,
"AE"));
                    DateTime ActualStart = DateTime.Parse(DumpTaskInfo(task, 0,
"AS"));
                    DateTime Today = DateTime.Now;

                    if (ActualStart <= Today && ActualEnd <= Today)
                    {
                        try
                        {
                            conn = new MySqlConnection(cs);
                            conn.Open();

                            MySqlCommand cmd = new MySqlCommand();
                            cmd.Connection = conn;

                            cmd.CommandText = "UPDATE `triggertable` SET
`Value`='1'  WHERE `ID`='1'";
                            cmd.CommandType = CommandType.Text;

                            cmd.Connection = conn;

                            cmd.ExecuteNonQuery();
```

```
//MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow, "Update");
                                DBUpdateDecisionTable(currentPhase.ToString(),
(currentActivity + 1).ToString(), "1");
                                currentActivity++;
                                currentPhase++;
                            }
                            catch (MySqlException ex)
                            {
                                MessageBox.Show("Error:" + ex.ToString());
                            }
                            finally
                            {
                                if (conn != null)
                                {
                                    conn.Close();
                                }
                            }

                        }
                    }
                    i++;
                }
            }
        }

        private string DumpTaskInfo(TimelinerTask task, UInt32 level, string Info)
        {
            for (Int32 i = 0; i < level; i++)
            {
                //Debug.Write("  ");
            }

            if (Info == "PS")
            {
                return task.PlannedStartDate.ToString();
            }
            if (Info == "PE")
            {
                return task.PlannedEndDate.ToString();
            }
            if (Info == "AS")
            {
                return task.ActualStartDate.ToString();
            }
            if (Info == "AE")
            {
                string AE = task.ActualEndDate.ToString();
                if (AE != "")
                {
                    return AE;
                }
                else
                {
                    return "";
                }
```

```
            }
            if (Info == "Name")
            {
                return task.DisplayName.ToString();
            }

            MessageBox.Show(task.DisplayName.ToString());
            //MessageBox.Show(builder.ToString());

            if (task.Children == null)
            {
                overLap = true;
            }
            else
            {
                overLap = false;
            }



            foreach (TimelinerTask child in task.Children)
            {
                DumpTaskInfo(child, level + 1, "");
            }
            return "";
        }

        private void facilityDetails_Click(object sender, EventArgs e)
        {
            FacilityDetail FacilityDetail = new FacilityDetail();
            FacilityDetail.Show();

        }

        public bool DBUpdateDecisionTable(string phase, string task, string value)
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "UPDATE decision SET `Within`='" + value + "'
WHERE Phase ='" + phase + "' AND Task ='" + task + "' ";
                cmd.CommandType = CommandType.Text;
                cmd.ExecuteNonQuery();
                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow,
"Error:" + ex.ToString());
                return false;
            }
```

```
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }

        }

        public bool DBCheckDecisionTable()
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "SELECT * from decision";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    if (dr[4].ToString() == "1" && dr[5].ToString() == "1")
                    {
                        DBUpdatekDecisionTable(dr[0].ToString(), "2");
                    }
                    else if (dr[4].ToString() == "1" || dr[5].ToString() == "1")
                    {
                        DBUpdatekDecisionTable(dr[0].ToString(), "1");
                    }
                }

                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }

        }

        public bool DBUpdatekDecisionTable(string ID, string Value)
        {
```

```
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "UPDATE decision SET Status='" + Value + "' WHERE
ID='" + ID + "'";
                cmd.CommandType = CommandType.Text;
                cmd.ExecuteNonQuery();
                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
        }

        public bool DBPhaseUpdate(string phase)
        {
            string cs =
"server=localhost;userid=root;password=hammad;database=sitelayout";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "UPDATE triggertable SET Value='" + phase + "'
WHERE ID='5'";
                cmd.CommandType = CommandType.Text;
                cmd.ExecuteNonQuery();
                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;
            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
```

```
                }
            }
        }

    }
}
```