UTILIZING REAL-TIME LOCATION DATA FOR PERFORMANCE MONITORING IN HEALTHCARE SYSTEMS

by

CENK DEMIR

A thesis submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Industrial & Systems Engineering

written under the direction of

Professor Thomas O. Boucher

and approved by

____

____

____

New Brunswick, New Jersey

October 2009

ABSTRACT OF THE THESIS

Utilizing Real-Time Location Data For Performance Monitoring In Healthcare Systems

By CENK DEMIR

Thesis Director:
Professor Thomas O. Boucher

Although both public and private healthcare expenditures have been very high in the US when compared to other wealthy and industrialized nations; the quality and reach of services provided have constantly been in dispute. For that reason, improving the efficiency of healthcare services has emerged to be an important goal. However, the unpredictable and complex nature of the healthcare environments makes this goal difficult to achieve. Recently industrial engineering methods started being applied to improve hospital efficiencies. In addition, the utilization of technological advancements such as RFID based real time location systems (RTLS) in the healthcare sector provides an additional opportunity to apply industrial engineering methods to healthcare.

In this thesis, a data transformation and analysis framework is developed to be employed as part of a RTLS schema in a hospital unit. This framework consists of a software agent that is capable of monitoring, analyzing and predicting the performance of a process from RTLS data. The software agent performs its task by means of several statistical methods customized for specific purposes. It can identify steady state behavior, changes in the mean and transient states such as learning curves. Its main purpose is to detect the effects of modifications on the system and forecast the future performance level. A simulation model is built to produce tracking data of entities in a hypothetical hospital unit to be fed into the software agent via a database structure. This completes the framework and enables the testing of the developed agent using realistic data.

RTLSs typically suffer from accuracy issues which result in imperfect tracking data. This makes performance monitoring very infeasible or even impossible. To overcome this issue, a data cleaning algorithm is developed which can be fully integrated with the developed performance agent. The algorithm utilizes a Bayesian approach in a sliding window analysis. Testing of the data cleaning algorithm is performed for various scenarios.

Acknowledgement

Thanks to my advisor and committee members.

Special thanks to my family.

Dedication

To my mother…

Table of Contents

List of tables

List of Figures

1. Introduction

In this chapter we present an overview of four different areas that constitute the main blocks of this thesis. Our study spans a wide range of concepts and techniques. It is based on continuous quality improvement (CQI) concept which utilizes quantitative methods to provide and maintain continuous quality evaluation and improvement especially in healthcare sector. We are motivated by the recent developments in radio frequency identification (RFID) based real time location systems (RTLS) which bring in opportunities like tracking of patients, staff and mobile equipment in a feasible way. These developments can be combined with various statistical techniques to devise new methods in the framework of CQI.

Simulation of hospital units and learning in healthcare sector are other notions that we incorporate in this research and introduce in this section. The utilization of real time location systems in a healthcare environment creates opportunities to use these ideas in a more effective way.

1.1. Continuous Quality Improvement

In the 1980s, the concept of Continuous Quality Improvement (CQI), which was already being utilized in the manufacturing sector, was introduced into the healthcare sector. CQI brought to the healthcare sector the idea that processes in a healthcare facility can and should be continuously evaluated and improved. [Langley, et al.].

CQI is a data driven approach which uses performance measures both as input and feedback. At the input step, performance measures are used to determine the current level of the quality of service. Evaluation is done to see if the current level of performance/quality is satisfactory. If it is not, a quality level is set as a goal and systematic incremental steps are taken towards achieving it. Feedback from the system is essential to monitor the changes in the performance and check if advancement towards the goal is acquired. This functioning of CQI requires continuous collection and monitoring of data. For this purpose, conventional statistical quality control charts like

Shewhart chart have been used. However, these charts require technical capability to set up and can only monitor a stationary process performance.

Data collection has always been very difficult in healthcare environments. In fast paced environments like emergency departments, keeping track of operations with manual methods is too interruptive to apply. Expecting the personnel to manually record performance parameters is a significant burden in such an environment. This is a prime difficulty of applying statistical methods in healthcare sector. Using CQI methods is not possible without collection of accurate data.

1.2. Simulation in Healthcare

Another reason to collect performance data is to derive probabilities and distributions for a simulation study. Simulation has long been used as a decision support tool in various sectors. There are several examples of simulation studies of healthcare facilities in the literature. Simulation is especially suited to the analysis of healthcare organizations due to its ability to handle high complexity and variability which is usually inherent in this sector. Experimentation of different workflows, staffing decisions and what-if analysis are all promising applications of simulation in healthcare. However, a typical simulation study requires deliberate data collection effort over a considerably long period of time. Sample studies in the literature suggest that a typical data collection period is at least a few months which is not only time consuming and labor intensive but also practically very infeasible in a healthcare environment.

1.3. Radio Frequency Identification

Although it has been available since the 1950s, Radio Frequency Identification (RFID) has gained new attention when it became more affordable in the past few years. The proposed ideas of usage extend to the point of automatic check outs in stores where a customer simply walks out of the store with the items she wants and a reader at the door automatically identifies the items and charges the price from her RFID credit card. While there are still several years until such a system can be deployed, RFID has already found application areas in supply chain and inventory management. Wal-Mart's request of attaching RFID tags to pallets and cases it gets from its top suppliers is a good example of how this new technology is finding its niche.

Collecting simulation data using RFID tracking technology has been used in supply chains and in very few healthcare environments. The potential of collecting large amounts of high quality data without any interruption to the system and using very little effort after the installation is especially promising for healthcare applications. RFID technology has already started being used in hospitals as a tool for locating mobile devices. There are a few companies providing this kind of system and several hospitals have adopted this technology to track their mobile assets. These developments show that RFID technology is feasible to be implemented in a hospital environment and there is an interest in this sector on the utilization of it. While this technology's compatibility in a hospital environment is still in question, successful applications are encouraging enough to further investigate it.

Collecting tracking data via RFID can also provide real-time performance data for CQI implementation. In a case where the performance measure to be monitored is the time required for a task to be completed, it is possible to link the RFID database to a monitoring software which can perform the desired analysis. This includes but is not limited to detecting changes in the performance and making predictions.

1.4. Learning

Learning by doing has been observed in various manufacturing systems. Understanding the factors that drive learning and promoting learning by organizational orientation have been major research topics. While the mechanics of learning are still not fully understood, its involvement in estimating the unit cost of a certain good in the long run or the future performance of a system is an important element of decision making, especially in the startup phase of a project. While learning at an individual level has been attributed most successfully to the number of units produced or number of times a service is given, learning at organizational level is still an open research subject. Learning in healthcare applications is one of our research interests in this thesis.

2. Objectives

Improving efficiency and quality of healthcare services has emerged to be a high priority task from both the healthcare providers' and patients' point of view. Increasing the utilization of resources and number of patients treated in a given time without compromising the quality of the service delivered has created conflicting objectives for healthcare providers. Maintaining and improving the quality of service while keeping the cost low is crucial for this sector.

There has been extensive effort and research based on the advances in technology to achieve these objectives. One of the major and important information technology innovations which have been introduced into hospitals and healthcare clinic settings to improve efficiency is the Real-Time Location System (RTLS). It combines Radio Frequency Identification (RFID) tags with tag readers and server software that locates the source of the tag signal. It has been used in several applications among which are tracking of critical mobile pieces of equipment, patients and staff members.

In this thesis we focus on methods that use data generated by RTLSs to monitor and improve hospital efficiency. More specifically, this thesis intends:

- To develop an intelligent software agent that monitors and evaluates a data stream for detecting change points and transient learning states, continuously resetting the control limits within the steady state region during each time epoch.
- To link the agent with a simulation model that realistically emulates an RTLS for equipment, patients, and staff and to develop methods that translate RTLS data into event data for defining performance metrics to be monitored and evaluated.

The contribution of this work is the design of the overall model. Namely:

1. Combining statistical methods in a single software agent model that is able track changes in performance metrics and identify transient states such as learning periods.

2. Developing an innovative method for estimating initial learning curve parameters to be used as input for a non-linear regression in order to fit a learning curve to non-steady state transient data.

3. Designing and implementing a procedure for translating RTLS data into event data that describes the evolution of the states of the hospital environment, thus providing useful statistical data for simulating that environment.

4. Designing and implementing a procedure for cleaning raw tracking data which may contain errors due to inaccuracies in the RTLS.

3. Literature Review

In this chapter we review the literature relevant to the proposed work of this thesis. The reviewed literature is grouped in four sections. In the first section examples of simulation studies in healthcare sector are given. In the second section the studies that relate to our utilization of learning curves are covered. The third section describes studies on methods for cleaning of RTLS data. The fourth section points out the studies on the statistical methods that are employed.

3.1. Simulation Studies

First of all, studies reviewed on data gathering and simulation in the healthcare environment are introduced. Simulation modeling has been widely used as a supportive tool for decision making in healthcare sector. The ability of handling large variations, complicated workflows and flexible "what-if?" analysis makes it especially useful in this context.

Ramakrishnan *et al.* (2004) use simulation to investigate the effects of implementing a digital image archiving system within the radiology department of Wilson Memorial Regional Medical Center. Performance measures used in the study are patient throughput and report generation time. Modified workflows are generated based on the current workflow to test different scenarios. Four months of data from Radiology Information Systems are studied and additional data is collected via time studies. Eight months of historical data are used for verification and validation.

Rossetti *et al.* (1999) use computer simulation to test alternative Emergency Department (ED) attending physician schedules and to analyze the corresponding impacts on patient throughput and resource utilization.

Weng and Houshmand (1999) perform a simulation study in a local healthcare clinic where key performance measures are patient throughput (number of patients served in unit time) and patient time (total time that a patient spends in the system). Different staffing scenarios aiming to improve these measures are investigated. Data collected over a two months time frame by benchmarking engineers are used but the data is sparse for fitting of certain distributions.

Miller *et al.* (2006) investigate data collection via RFID technology. They outline the advantages and disadvantages of collecting simulation data via RFID based on their experience at an Emergency Department. Infrared sensors, which require direct line of sight with the receivers, are used to track patients. Although this method is promising in terms of the amount and quality of the data, technical inefficiencies and human-based issues require a well planned execution. Main issues they have are covered and lost tags, misreadings, database access problems and reluctant, untrained personnel.

Amini *et al.* (2007) perform a simulation study in an emergency department using RFID generated data. Data for seven hundred patients are collected over a four week period. Locations that a patient visits are mapped to events in the flowchart of the model after a data cleansing process and the time that the patient spends in that location are taken as the time duration of the event. Probability distributions for simulation are derived from this transformed data. RFID generated data is found to be particularly useful in associating the time spent by each patient to events. The percentage of the time that can be explained with RFID generated data is significantly higher than the percentage of the time that can be explained with manually collected data. Specifically, with the manually collected data, only 25% of an average patient’s stay can be accounted for. On the other hand, using RFID data, 80% of a patient’s stay can be accounted for. As a result, it is concluded that it is feasible to use RFID technology as a data collection method for simulations of healthcare systems. As a future research topic, it is suggested to develop more efficient heuristics and algorithms to process and manipulate data from various sources.

Another study on gathering RFID data by using real-time tracking technology is conducted by Isken *et al.* (2005). Infrared technology is used in this study, where direct line of sight between the tag and the receiver is not necessary but the tags and receivers still need to be uncovered. Data cleansing steps were explained in detail.

3.2. Learning Curves

Learning curves have been explored mainly in manufacturing industry. Nevertheless there are also studies on learning in the service sector, including healthcare.

Ramsay *et al.* (2000) do a systematic literature review of learning curves in health technologies. It is stated in the paper that performance of many nondrug technologies change over time and further technical development may continue after introduction of a new technology. This phenomenon is commonly described as a learning curve. It is also concluded that currently used statistical methods in analysis of learning curves are not rigorous enough. Many of the studies that are reviewed rely upon descriptive data to show that learning has taken place without any formal statistical testing. It is noted that methods that can estimate the rate and length of learning together with the final skill level are needed.

Baloff (1971) conducts an empirical study of learning curves in three different industries. Baloff aims to extend the learning curve literature beyond aerospace industry by investigating production data of a musical instruments manufacturer, an apparel manufacturer and an automobile assembler. The conventional learning model given as $Y = ax^{-b}$ is used and the data is fit by employing simple linear regression after linearizing the data by taking logarithms. It is also demonstrated that for some of the datasets there is a distinct steady state phase occurring after learning. It is concluded in the research that it is risky to assume that there will be no steady state behavior in the learning curve.

Danford *et al.* (1995) work with learning curves to analyze data for pediatric radiofrequency ablation. A negative exponential learning model is used given as: $Outcome = a + bexp(-c * experience)$ where the outcome is taken as success rate, complication rate, mean procedure time and mean fluoroscopy time one at a time. Learning models, which are fitted via nonlinear regression, are used to determine the milestones of learning and estimate the steady state performance level. High R^2 values are obtained, which show that the trend in the data can be strongly explained by the learning equation.

3.3. Data Cleaning

Elnahrawy and Nath (2003) present a Bayesian method to reduce the uncertainty in a sensor network, which arises due to random noise. This method combines the prior knowledge of the true reading, error characteristics of the sensor, and the observed noisy reading in a Bayesian framework.

Jeffery *et al.* (2006) develop an adaptive smoothing filter for RFID data cleaning. In the system considered, each RFID reading is either true or false and indicates the presence of a tag at the location monitored. A sliding window procedure is employed where the window size is adapted dynamically to provide both completeness and accurate detection of transitions. Adaptation of the window size is achieved by modeling of observed RFID readings as an unequal-probability random sample of tags in the physical world.

3.4. Statistical Techniques Used

Marquardt (1963) develops an algorithm for the least-squares estimation of nonlinear parameters that interpolates between Gauss-Newton method and gradient descent method. The author states that the algorithm inherits the ability to converge from an initial guess which may be outside the region of convergence of other methods from gradient descent and the ability to close in on the converged values rapidly after the vicinity of the converged values has been reached from Gauss-Newton method.

Fahmy and Elsayed (2005) develop a new approach to detect linear trends in a process mean. This new approach makes use of the deviation between the target mean and expected mean for a moving window. It is shown that the square of this deviation follows a chi-square distribution with one degree of freedom. The proposed approach is compared with cumulative sum, Shewhart, generalized likelihood ratio and exponentially weighted moving average charts. This method is found to be able to deal with a wide range of trend magnitudes effectively.

Hudson (1966) conducts a study of multiple segment regression. Several cases are considered in which two or more sub models are fitted to data where the locations of joining points may or may not be known, using least square estimates. The simplest case is when the number of sub models and the location of the joining points are known. In this case the solution is obtained by finding the estimators of model parameters by minimizing the sum of square errors for each sub model. When locations of the joining points are not known, the problem becomes an exhaustive iteration, where the combination of joining points that minimizes the total sum of square errors of the sub models gives the best estimators for the parameters of the sub models. Note that this means that every possible combination of joining points has to be tried. A similar idea is used in the two part fit algorithm in post change point analysis as a part of this thesis.

4. Theory

In this chapter we present the theoretical foundations of our research. The statistical methods that we define here are combined in our Methodology chapter in order to achieve the proposed goals.

4.1. Portmanteau Test

A Portmanteau test tests whether the residuals of a data series are independently and identically distributed (iid) or not. The null hypothesis is that none of the autocorrelation coefficients up to a specified lag are different from zero. If null hypothesis is not rejected, it can be concluded that there is no significant autocorrelation in the data and a probability distribution can be fit.

Lag-h sample autocorrelation for a data stream is calculated as:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \qquad -n < h < n \tag{4.1}$$

where $\hat{\gamma}(h)$is the sample autocovariance function calculated as:

$$\hat{\gamma}(h) = n^{-1} \sum_{t=1}^{n-|h|} \left(x_{t+|h|} - \bar{x}\right)(x_t - \bar{x}) \qquad -n < h < n \tag{4.2}$$

where

n is the number of data points in the sample,

$\bar{x}$ is the calculated mean of the sample data.

For large n, the sample autocorrelation coefficients of an iid data stream with a finite variance can be approximated as iid with distribution $N(0, 1/n)$. Therefore, the sum of h sample autocorrelation coefficients is distributed as a chi-square distribution with h degrees of freedom. From this observation, the portmanteau statistic is calculated as:

$$Q = n \sum_{j=1}^{h} \hat{\rho}^2(j) \tag{4.3}$$

If $Q > X_{1-\alpha}^2(h)$, where $X_{1-\alpha}^2(h)$ is the chi-square statistic with h degrees of freedom and confidence level α, the null hypothesis is rejected. Otherwise, it is concluded that the sample data does not have significant autocorrelation with given confidence.

A refinement of this statistic is suggested by Ljung and Box which works better especially for small sample sizes. The refined statistic, which is also used in our agent, is calculated as:

$$Q = n(n+2)\sum_{j=1}^{h} \hat{\rho}^2(j)/(n-j) \tag{4.4}$$

4.2. Chi Square Change Point Detection Method

The Change point detection method that is used in this research was proposed by Fahmy and Elsayed in 2005. It utilizes a moving window procedure and issues an out-of-control signal when the statistic for a certain window exceeds the upper control limit. Let the in-control process be distributed as:

$$y = N(\mu_0, \sigma_0^2) \tag{4.5}$$

For a window of size w, let

$$\hat{\mu}_w = \hat{\alpha} + \hat{\beta} x_w \tag{4.6}$$

be the expected process mean at the end of the window where $\hat{\alpha}$ and $\hat{\beta}$ are the coefficients of the linear model fitted to the window by least squares estimation. The difference between the actual and fitted mean is distributed as follows:

$$(\mu_0 - \hat{\mu}_w) = N(0, \sigma_0^2(\frac{1}{w} + \frac{(x_w - \bar{x})^2}{S_{xx}})) \tag{4.7}$$

where

$$S_{xx} = \sum_{i=1}^{w} (x_i - \bar{x})^2 \tag{4.8}$$

By normalizing and taking the square we obtain a chi-square distribution with one degree of freedom:

$$\left[\frac{(\mu_0-\hat{\mu}_w)}{\sqrt{\sigma_0^2\left(\frac{1}{w}+\frac{(x_w-\bar{x})^2}{S_{xx}}\right)}}\right]^2 \sim X_1^2 \tag{4.9}$$

The upper control limit of the statistic is calculated as:

$$P(X_1^2 > UCL \middle| \mu_x = \mu_0) = \alpha \tag{4.10}$$

where α is the probability of type-1 error. Based on the convention of taking in-control average run length as 370,

$$\alpha = \frac{1}{370} = 0.0027 \tag{4.11}$$

From Chi-Square table, with 1 degree of freedom, UCL is found to be 8.99. While this UCL is accurate for a small window size, calibration is necessary if a larger window size is to be used. This is mostly due to the high correlation between consecutive window statistics. As a result of their simulation study to find more precise UCL for given ARL of 370, Fahmy and Elsayed propose using UCL = 8.99 for w=3, UCL=8.85 for w=5, UCL=7.65 for w=20.

Once an out-of–control signal is issued, the change point is located by counting backwards through the monotonically increasing series of statistics of the consecutive windows. The last data point of the window whose statistic is the first element of the monotonically increasing series is identified as the change point.

4.3. Learning Models

Learning models that are used in this study have steady state components. While some conventional equations used to model learning do not include a steady state component, some studies provide strong evidence that steady state performance is a result of learning process [Baloff, 1971]. Specifically, the models that are used:

$$Y_x = a + bexp(-cx) \tag{4.12}$$

$$Y_x = a + bx^{-c} \tag{4.13}$$

where

Y_x is the performance measure,

a is the steady state performance,

x is experience measured in number of times the task was performed,

$bexp(-cx)$ and bx^{-c} are the transient parts of the learning equations.

4.4. Nonlinear Regression

Fitting a set of data to a nonlinear learning model is performed using nonlinear regression techniques. Levenberg-Marquardt method is used for this purpose. This method combines the power of two nonlinear regression techniques, namely Gauss-Newton algorithm and gradient descent. While gradient descent method is very efficient in the initial steps of the iteration, it provides linear convergence when the search is performed in a close neighborhood of local minimum, thus leading to very slow convergence. On the other hand, Gauss-Newton method is very effective in the final stages of convergence. The algorithm developed by Marquardt interpolates between these two methods by utilizing a damping factor. The algorithm, as it is used in this thesis is given below:

Let $\boldsymbol{y} = [y_1, y_2, \dots, y_n]^T$ be the vector of dependent variables and $\boldsymbol{x} = [x_1, x_2, \dots, x_n]^T$ be the vector of independent variables of a dataset. Let $y_i = f(x_i, \boldsymbol{a})$ be the nonlinear model that is wanted to fit to the data where $\boldsymbol{a} = [a_1, a_2, \dots, a_k]^T$ is the vector of model parameters that are wanted to calculate so that sum of square errors, $\sum_{i=1}^{n}(y_i - f(x_i, \boldsymbol{a}))^2$ is minimized. Let $\boldsymbol{d} = [d_1, d_2, \ldots, d_n]^T = [y_1 - f(x_1, \boldsymbol{a}), y_2 - f(x_2, \boldsymbol{a}), \ldots, y_n - f(x_n, \boldsymbol{a})]^T$ be the vector of residuals and

$$J = \begin{bmatrix} \frac{\partial d_1}{\partial a_1} & \cdots & \frac{\partial d_1}{\partial a_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial d_n}{\partial a_1} & \cdots & \frac{\partial d_n}{\partial a_k} \end{bmatrix}$$

be the Jacobian matrix.

1. Set maximum number of iterations, sensitivity and λ to desired values, say 1000, 0.001 and 0.01 respectively. Set m=1.

2. Set initial guesses for the parameters, $\boldsymbol{a}_m$.

3. Calculate J and $\boldsymbol{d}_m$.

4. Solve the set of linear equations $(J^T J + \lambda I)\boldsymbol{h}_m = -J^T \boldsymbol{d}_m$ for $\boldsymbol{h}_m$.

   This is the crucial step where the algorithm takes a step, $\boldsymbol{h}_m$. Note that λ is the damping factor in the equation. For large values of λ, the algorithm takes a short step in the steepest descent direction while for small values, it mimics the Gauss-Newton method taking a good step in the final stages of iteration.

5. Calculate new set of parameters $\boldsymbol{a}_{m+1} = \boldsymbol{a}_m + \boldsymbol{h}_m$.

6. Calculate $\boldsymbol{d}_{m+1}$.

   Case 1: If $\boldsymbol{d}_{m+1} < \boldsymbol{d}_m$ and $\frac{\boldsymbol{d}_m - \boldsymbol{d}_{m+1}}{\boldsymbol{d}_m} < sensitivity$, $\boldsymbol{a}_{m+1}$is the vector of least square estimates of model parameters. End of iterations.

Case 2: If $\boldsymbol{d}_{m+1} < \boldsymbol{d}_m$ and $\frac{\boldsymbol{d}_m - \boldsymbol{d}_{m+1}}{\boldsymbol{d}_m} \geq sensitivity$, divide λ by 10 , increment m by 1, go to step 3.

Case 3: If $\boldsymbol{d}_{m+1} \geq \boldsymbol{d}_m$, multiply $\lambda\ by\ 10$, increment m by 1, go to step 4.

4.5. Estimating Starting Parameters

Although Levenberg-Marquardt algorithm provides a solid means of calculating estimates for parameters in the learning models, convergence still depends heavily on the initial guesses. A bad initial guess can easily result in divergence or convergence to an undesired local minimum. The selection of initial parameters are generally done by observation in iterative techniques; however in this case this is not possible since the aim is to build a system that does not need any user input to the analysis. At this point a robust method is developed to calculate initial estimates for the learning curve parameters. The method is given below:

1. Fit the conventional learning curve to the data, which is given as:

$$Y = kx^c \tag{4.14}$$

The sub-steps are:

a. Take the logarithm of the data and the model. The new learning model is :

$$\log Y = \log k + c \log x + \epsilon \tag{4.15}$$

Where ϵ is the error term.

b. Perform simple linear regression.

c. Take exponentials. At this stage $\hat{k}$ and $\hat{c}$ have been calculated.

2. Estimate a by searching for “steady state like” behavior on the learning curve. Using the model $\hat{Y} = \hat{k}x^{\hat{c}}$ look for a region where the percentage decrease in the dependent variable

is below a certain number, say 0.0005. This can be accomplished in the following fashion:

$$-\frac{\frac{dy}{dx}}{y} < 0.0005 \qquad (4.16)$$

$$\frac{\hat{c}\hat{k}x^{\hat{c}-1}}{\hat{k}x^{\hat{c}}} > -0.0005 \qquad (4.17)$$

$$x > -\frac{\hat{c}}{0.0005} \qquad (4.18)$$

Plugging the derived x into the learning curve equation gives an estimate for the steady state component a.

3. Equating Eq. 4.14 and Eq. 4.13 for $x = 1$ gives:

$$\hat{b} = \hat{k} - \hat{a} \qquad (4.19)$$

For $x = 2$;

$$\hat{c} = \log_2(\hat{y}_2 - \hat{a})/\hat{b} \qquad (4.20)$$

Where $\hat{y}_2$ is the estimation of the dependent variable calculated from Eq. 4.14.

Thus, we have calculated the initial estimates for Eq. 4.13.

4. For $x = 1$ and $x = 2$ Eq. 4.12 gives:

$$e^{\hat{c}} = \frac{\hat{y}_1 - \hat{a}}{\hat{b}} \qquad (4.21)$$

and

$$e^{2\hat{c}} = \frac{\hat{y}_2 - \hat{a}}{\hat{b}} \qquad (4.22)$$

It is trivial to derive;

$$\hat{b} = \frac{(\hat{y}_1 - \hat{a})^2}{(\hat{y}_2 - \hat{a})} \tag{4.23}$$

and

$$\hat{c} = \log(\frac{\hat{y}_1 - \hat{a}}{\hat{b}}) \tag{4.24}$$

Thus, the initial estimates for Eq. 4.12 have been calculated.

4.6. Durbin-Watson Test

The Durbin-Watson test tests the null hypothesis that the residuals from a least-squares regression are not auto correlated. It is calculated as,

$$d = \frac{\sum_{i=2}^{n}(e_i - e_{i-1})^2}{\sum_{i=1}^{n} e_i^2} \tag{4.25}$$

where n is the number of data points and,

e_i is the residual term of the i^{th} data point.

Durbin-Watson statistic takes a value between 0 and 4 where 2 indicates no auto correlation. A Durbin-Watson statistic larger than 2 is an indicator of negative autocorrelation while a statistic smaller than 2 is an indicator of positive autocorrelation. Tables for the critical values of Durbin-Watson statistics have been established. A Durbin-Watson statistic table lists lower and upper critical values, $d_{L,\alpha}$ and $d_{U,\alpha}$for combinations of number of regression parameters and sample size. To test for positive auto correlation, the test statistic d is compared to associated the $d_{L,\alpha}$ and $d_{U,\alpha}$ where α is the significance.

- If $d > d_{U,\alpha}$ do not reject null hypothesis that autocorrelation is zero.
- If $d < d_{L,\alpha}$ reject null hypothesis. Residuals are auto correlated.
- If $d_{L,\alpha} < d < d_{U,\alpha}$ the test is inconclusive.

The test of negative autocorrelation can be performed by using the statistic $(4 - d)$ in the same manner.

5. Methodology

A real-time performance monitoring software agent is developed where the input is the performance metric of a continuously repeated process. The time spent for completion of a certain process is a typical example of the performance criterion that is to be monitored. The system of interest, the process, the performance metric and the data collection method may vary; however this research is primarily based on healthcare systems where data is collected via an RFID tracking system. Specifically, the software agent recognizes steady state periods and calculates distribution parameters of the performance metric, uses these parameters to detect change points in the mean, identifies the nature of this change, fits a learning model in cases where learning is observed in order to make predictions on the future steady state performance and looks for regions to determine a new steady state level. This agent does not need any user input as starting or calibration parameters. It has the capability to adapt to new performance levels and perform analysis by itself. It provides necessary statistics which may aid the decision maker. Its role in a continuous quality improvement framework is to detect the effects of the modifications done to the system.

As an extension to this task, data parsing algorithms used in transforming of real-time tracking data to probability distribution data, which can be used for performance monitoring or simulation analysis, are investigated. Note that this is necessary to show how the developed monitoring tool fits into the framework of Real Time Location Systems.

5.1. Performance Monitoring

In CQI perspective, it is essential to get a measurable feedback when a new method or technology is introduced to the system. It is assumed in this research that without interruptions, a system performs in steady state. When a new method or technology is introduced to the system, it is likely to observe a detectable change in the performance metric being monitored. The performance metric may be directly or indirectly related to the change in the system. For example, an improvement in a laboratory test can lead to a reduction in the total time spent by a patient in the facility. This reduction in time follows a learning of the new process by laboratory technicians and the learning results in better system performance. Modeling of the learning curve after the change point can give an estimate on the future steady state level of the performance, thus forecasting the possible outcomes of the modification done. Continuous monitoring of the process increases the quality of the forecast and leads to a new steady state period where the new distribution parameters can be estimated. The performance monitoring agent developed in this thesis is composed of different statistical methods all of which are optimized and combined for this specific purpose. The workflow logic of this agent is summarized in the Figure 5.1.

The agent monitors the data stream continuously. After each new data point arrives, the state of monitoring is checked and necessary calculations are done for a data window with a pre determined size where the last data point in the stream is the also the last data point in the window. Three possible states are: Start up State, Steady State, and Post Change Point State.

5.1.1. Start Up State

When the agent is launched, it is in the “Start up” state where the parameters of the process monitored are not known. So the first task is to check if a steady state performance level is attained. For this purpose, a moving window of size 30 or 50 is used and two different statistical tests are applied. If enough data points do not accumulate, the agent waits until more data points are read to perform the analysis.

The performance of a system when it is in control is generally modeled as a normal distribution in the literature. This gives the opportunity to use a wide array of change point detection methods. In order to fit a normal distribution to a data window, the agent makes sure that two criteria are met:

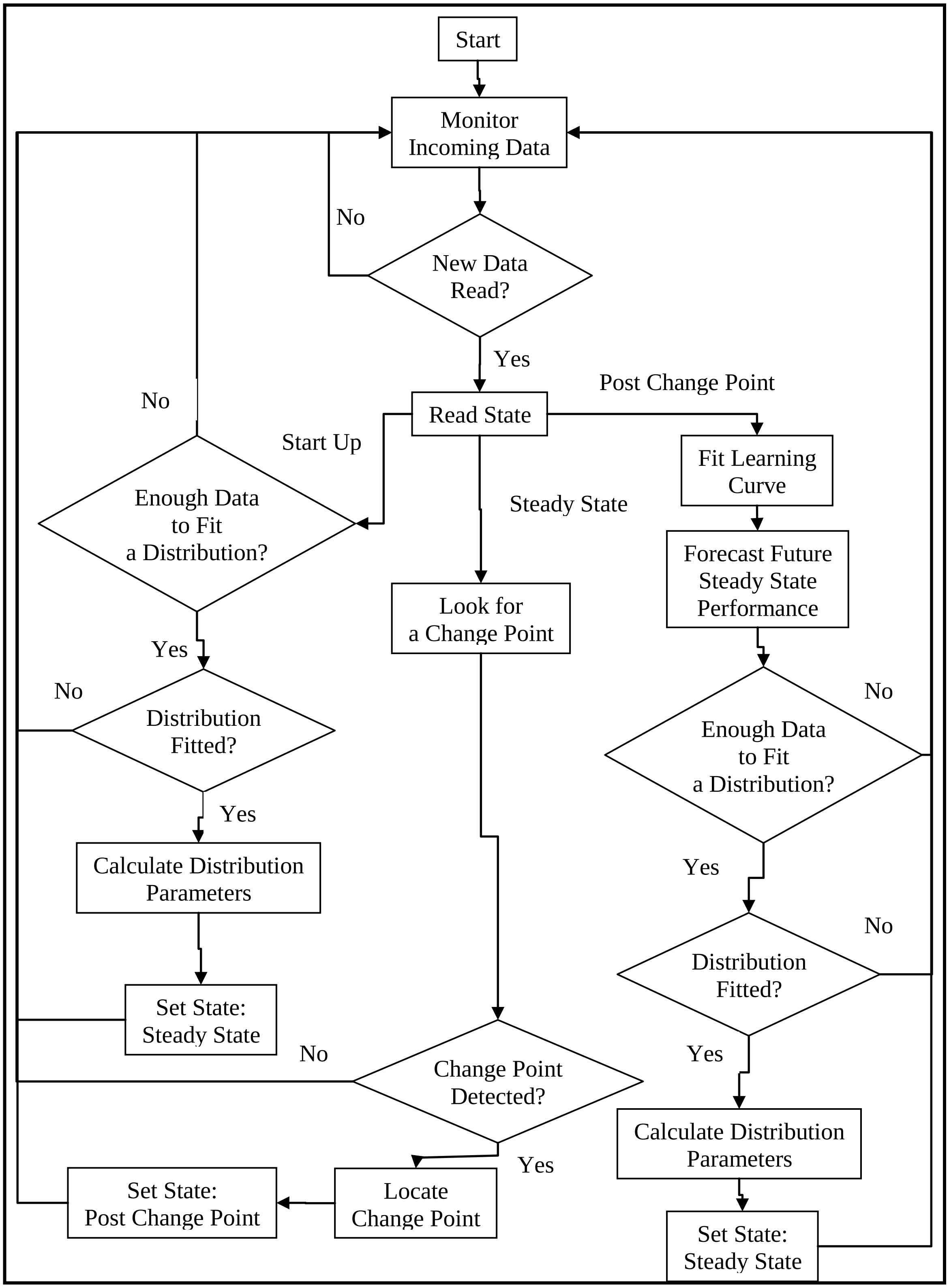


Figure 5.1 Logic Flowchart of the Software Agent

1) The data stream is independently and identically distributed.

2) The data stream passes a chi-square distribution fit test.

The first criterion is essential for the assumption of steady-state behavior. A portmanteau test is used to check if lag-h (h=1,2,…,10) autocorrelations are different from zero for the given window. If enough evidence cannot be found supporting the hypothesis that lag-h (h=1,2,…,10) autocorrelations are different from zero, no further analysis is performed for the current window and the agent continues waiting for a new data point.

If the current window passes the portmanteau test, it is concluded that there is no significant autocorrelation in the data and a chi square test is performed. If the null hypothesis that the sample data comes from a normal distribution with the calculated mean and standard distribution cannot be rejected at a given confidence level, the agent concludes that the process is in control. The calculated mean and standard deviation are recorded as process parameters and the state of the watch is changed to “Steady State” where the agent monitors the process to detect a change in the performance.

5.1.2. Steady State

When the process is in steady state the agent uses the pre-determined process parameters to detect an out-of-control signal. This is a typical change point detection problem encountered frequently in statistical quality control.

While Shewhart and CUSUM charts have been utilized widely in statistical quality control, a relatively new method is used in this research to detect changes. This method, which is going to be referred as Chi-Square method hereafter, is proposed by Fahmy and Elsayed (2005) and uses a window procedure to search for change points. It does not require any parameters to be set for detection unlike (for instance) CUSUM where two different parameters (k, the threshold which ignores small deviations and h, the upper limit for the statistic) need to be set for a given mean, standard deviation and the magnitude of the change desired to be detected. This new approach is not only very handy in practical applications but is also powerful enough to be used instead of these mentioned conventional charts [Fahmy, 2005].

A window size of 20 is used to detect a change point. After arrival of each data point, the agent runs the change point detection algorithm for the last 20 data point window. If a positive signal is given by the calculation, the change point is located using the method proposed by Fahmy and Elsayed (2005) and the system state is updated as “Post change point”. If a negative signal is given, the agent waits for more data.

5.1.3. Post Change Point State

Detection of a change point may have different reasons. It is not possible to determine the reason or validity of a change in a process without making a physical inspection of the process. Also, for any change point detection schema, exactly identifying the change point is not possible due to the randomness of the data. The detected change point may be an earlier or later point in the stream. Or even worse, it may be a false alarm when there is no change in the monitored process parameters. However, the proposed agent has to continue the analysis in all cases so that the analysis to be performed after a change point is detected should be able to handle any situation.

To accomplish this task, the agent tries to match the behavior of the process with one of the predetermined scenarios. The scenarios that are considered are given in Figure 5.2, Figure 5.3 and Figure 5.4.

In order to select the most likely scenario, the agent takes into account R^2, sum of square errors, Durbin-Watson statistic and the transient term of the fitted learning model. The logic flowchart of matching the data with the most likely scenario is given in Figure 5.5.

- The change is followed by a shift in the mean.

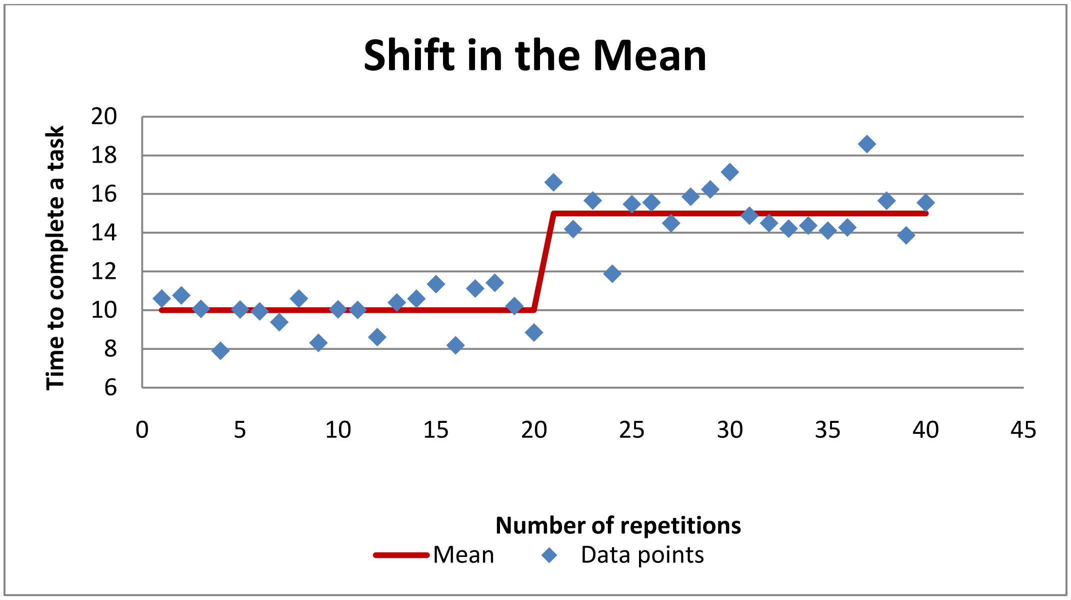


Figure 5.2 Step shift in the mean of a process

- The change is followed by immediate learning.

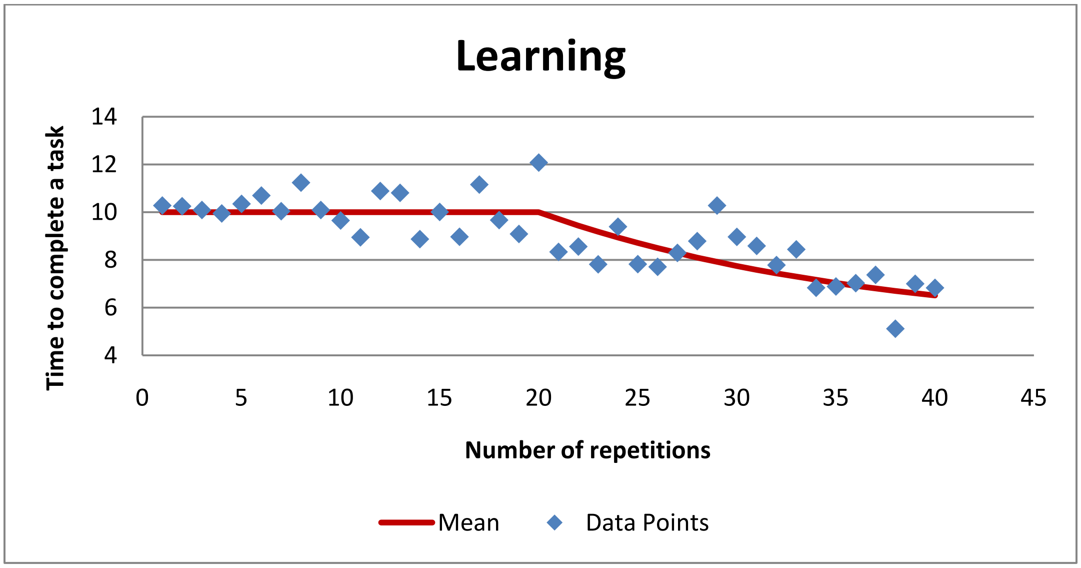


Figure 5.3 Immediate learning after the change in performance

- The change is followed by a shift in the mean, which is followed by learning.

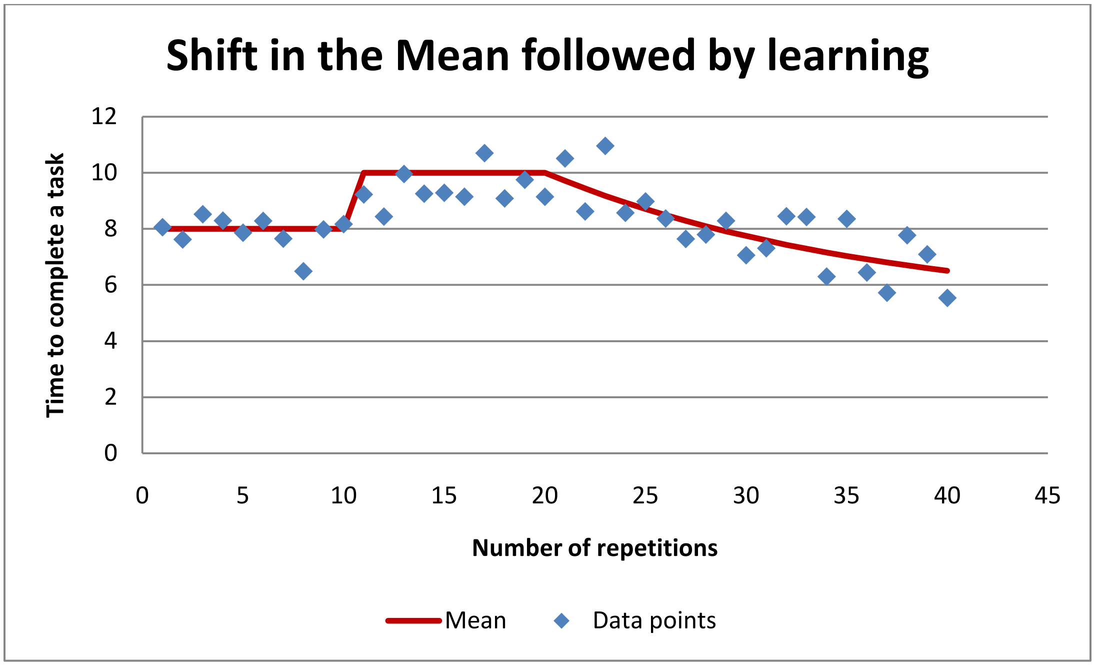


Figure 5.4 Shift in the mean followed by learning

Detecting a case where a shift in the mean is followed by learning is an example of two phase regression when the steady state period after the change is not long enough to establish new distribution parameters. If this transition state is long enough, the agent sets up new distribution parameters and treats the beginning of the learning curve as a new change point. On the other hand, when this period is not long enough, the agent should be able to determine the actual point where learning starts in order to better fit a model and make a reasonably accurate forecast.

Otherwise, all the data points after the initial change will be considered as on the learning curve which may lead to bad fit, poor forecast or no convergence.

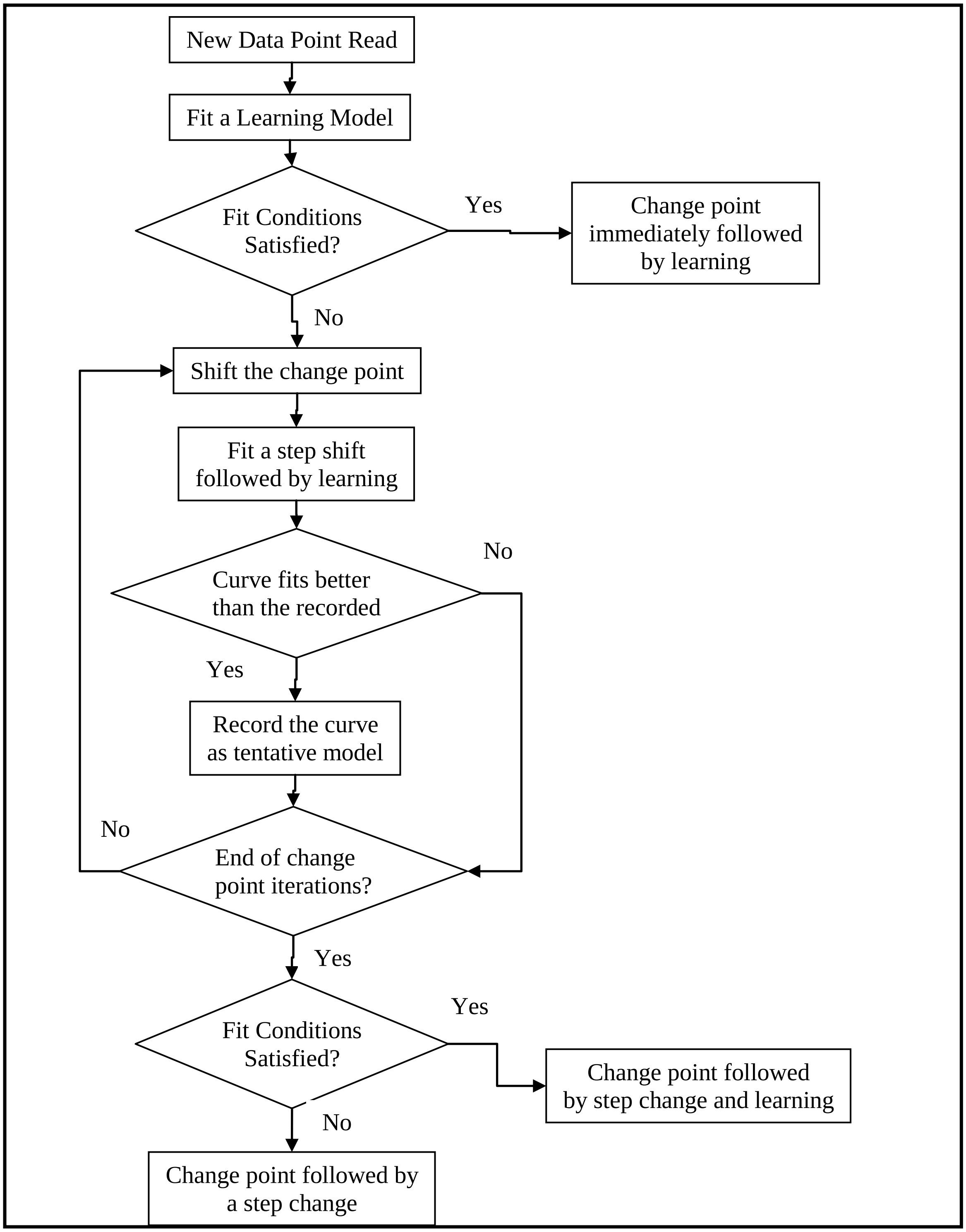

New Data Point Read
Fit a Learning Model
Fit Conditions Satisfied?
Yes
Change point immediately followed by learning
No
Shift the change point
Fit a step shift followed by learning
Curve fits better than the recorded
No
Yes
Record the curve as tentative model
No
End of change point iterations?
Yes
Fit Conditions Satisfied?
Yes
Change point followed by step change and learning
No
Change point followed by a step change

Figure 5.5 Curve fitting in post change point analysis

When new data is read, the agent tries to fit a learning model to the subset of data starting with the change point, provided that enough points have accumulated. If the fitted curve satisfies predetermined conditions of fit, it is accepted as an accurate representation of the data and the results are reported. If the fitted curve cannot satisfy all of the required conditions, the agent begins two part fitting algorithm. This algorithm breaks down the data stream into two parts and fits a steady state process to the first part while fitting a learning model to the second. All of the possible combinations are considered and the best fit is selected depending on the combined sum of square errors of the two parts. When the algorithm exits, the fitted learning model is evaluated based on the predetermined conditions. If these conditions are met, two part fit is accepted as a good representation of the process and the results are reported. Otherwise, the agent concludes that there is no learning taking place and the change in the process is caused by a shift in the mean. Note that a false alarm is a special case of mean shift where the magnitude of the shift is zero and can be treated in that manner.

The conditions of fit are set heuristically since there is no formal way of evaluating the absolute quality of fit for any regression analysis. A learning curve is accepted as a good representation of the process if all of the following criteria are met:

1. R^2 is larger than a threshold: This criterion makes sure that the model is successful in explaining the deviation in the data. The threshold value should be chosen such that rate of rejecting a learning curve when one actually exists is low. Since there is no strict guideline of finding a threshold which is always guaranteed to work, this value should be set by experimentation. Currently the agent uses 0.2 as the threshold value for R^2.
2. The model passes the Durbin-Watson test: Failure to pass the Durbin-Watson test is an indication of autocorrelation in residuals of the fitted model, thus, is also an indication of poor fit.
3. The process model is not dominated by the steady state term: From our experimentations with different synthetic datasets, it was observed that the transient term of the learning equation tends to converge to zero if the equation is fitted to a stationary distribution.

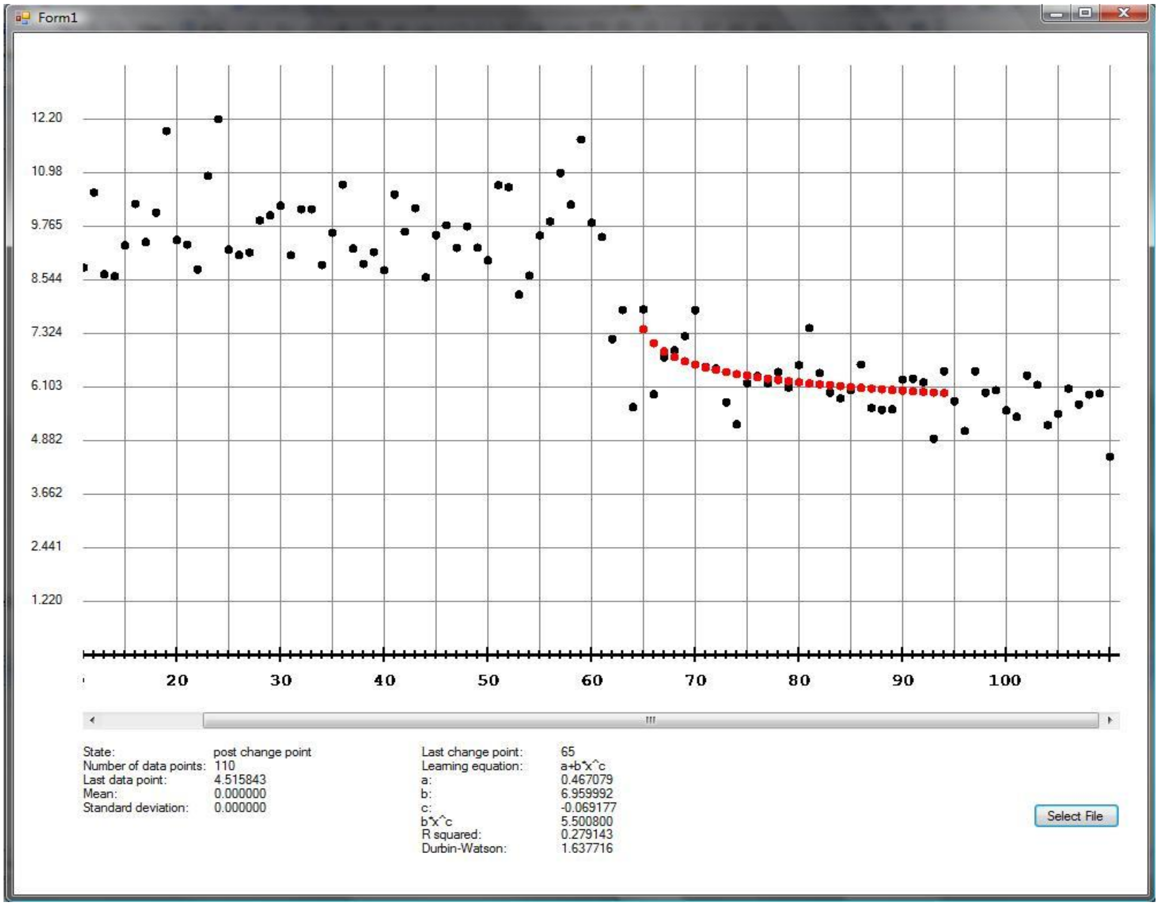
> This is a result of the fact that the best fit to a stationary distribution is a straight line. This observation is used such that a fitted model is rejected if the ratio of the transient term to the steady state term is below a predetermined threshold. Currently 0.001 is used as the threshold.

If a learning equation can be fitted to the data, the agent reports the parameters of the fitted curve, R^2 value and Durbin-Watson statistics. At the same time, the agent checks if enough data points are accumulated to establish a new steady state. When enough data points are accumulated, a new distribution fit test is performed along with the portmanteau test to check if a new steady state period is reached and the fitted learning curve (therefore the forecast too) is updated. Once a distribution is fitted to the data stream, the agent switches to “steady state” monitoring and starts looking for a new change point.

5.2. Programming

C# is employed as the development language because of its strong graphical user interface support and debugging options.

A graphical user interface is utilized in order to aid the decision maker in visually understanding and evaluating the data and the analysis. Once linked to a data source, the agent starts reading the file at fixed time intervals and performs analysis if new data is detected. Each data point triggers new calculations based on the state variable and is plotted on the graph. Certain statistics are shown on the screen as well as written in an output file. The graph can adapt to new levels of performance by scaling the data. Fitted learning curves are also plotted on the graph. Note that once the input file is linked to the agent, no operator assistance is necessary for the agent to function, as proposed.



State: post change point
Number of data points: 110
Last data point: 4.515843
Mean: 0.000000
Standard deviation: 0.000000

Last change point: 65
Learning equation: a+b*x^c
a: 0.467079
b: 6.959992
c: -0.069177
b*x^c 5.500800
R squared: 0.279143
Durbin-Watson: 1.637716

Figure 5.6 User interface of the software agent

5.3. Simulation

Based on the literature survey on real-time tracking systems and simulation in healthcare systems, a simulation model of a hypothetical hospital unit is built. This simulation model acts as an element of the continuous quality improvement framework by integrating with the software agent developed via a database structure. This framework enables the researcher to mimic a real-life system and test the software agent with certain scenarios. For this purpose, the simulation model must successfully emulate both a hospital environment and a real-time tracking system. The simulation model produces synthetic tracking data to be analyzed with the developed software agent.

5.3.1. Description of the Clinic

The simulated hospital unit is based on an outpatient clinic which serves patients between established hours. It has a well defined workflow where all the patients go through a combination of processes. It is assumed that patients, doctors, nurses and critical mobile equipment are monitored by a RTLS. Hereafter, these are referred to using the simulation term “entity”.

An arriving patient first checks in at the register where she is assigned an RFID tag. Then she proceeds to the waiting room to wait for an available consultation room. When a consultation room is available, the patient proceeds into it while a doctor and a nurse are called to that room. If a doctor or a nurse is not available, then the other entities wait in the consultation room until the missing one arrives. Arrival of the last entity marks the start of consultation.

After the consultation is complete, both the nurse and the doctor return back to their rooms. The consultation can have two results: Either the patient is sent to treatment or she is sent back to the register to leave the unit. This patient returns to the register, leaves her tag and exits the system. The others return back to the waiting room to wait for an available treatment room. When a treatment room is available, the patient proceeds into the treatment room while a nurse and a mobile equipment are called. If a nurse or a piece of equipment is not available, other entities wait in the treatment room until the missing one arrives. The arrival of the last entity marks the start of treatment set-up. After the set-up operation is complete, the nurse leaves the room and returns to the nurses’ room while the patient starts getting the treatment via the mobile equipment. When the treatment ends, the mobile equipment is taken to the cleaning room to be cleaned for next use and the patient returns to the register to leave her tag and check out.

The process flowcharts for each of the entity are given in Figure 5.7.

The doctors work in their office unless they are called for a consultation. If there is no doctor available when a call for consultation is received, the next available doctor heads to the room which has sent the earliest request. This is a simplification because in a real situation, doctors have assigned patients. However, this simplification is acceptable for generation of synthetic data for the monitoring agent.

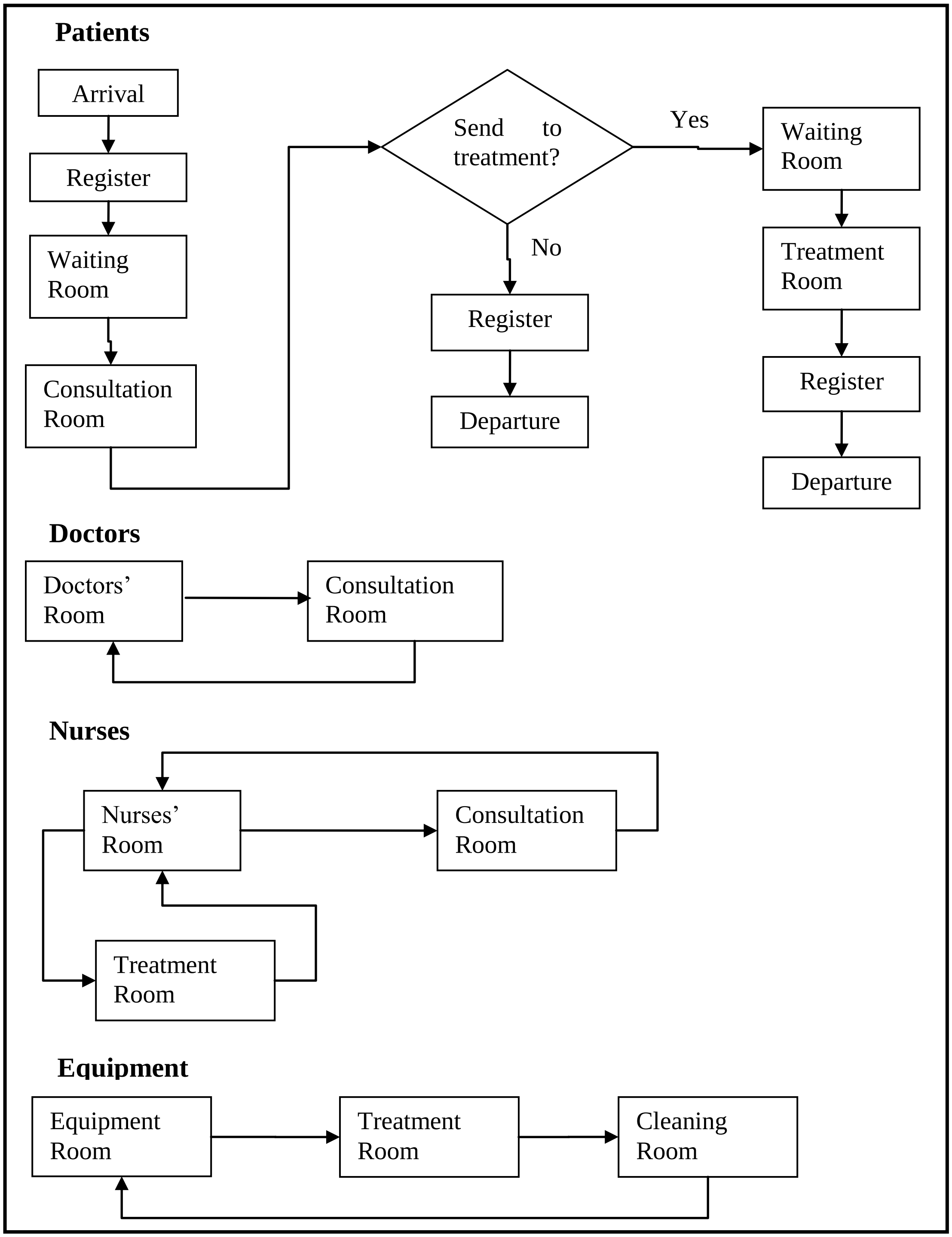


Figure 5.7 Entity flowcharts for the simulation model

Table 5.1 Event Mapping List

Event	Location	Entities
Consultation	Consultation Room	Patient Doctor Nurse
Treatment Set Up	Treatment Room	Patient Nurse Equipment
Treatment	Treatment Room	Patient Equipment
Waiting	Waiting Room	Patient
Check-In, Check-Out	Register	Patient
Equipment Cleaning	Equipment Cleaning Room	Equipment
Nurse Idle	Nurses’ Room	Nurse
Doctor Idle	Doctors’ Room	Doctor
Equipment Idle	Equipment Room	Equipment
Waiting	Consultation Room	Patient
Waiting	Consultation Room	Patient Nurse
Waiting	Consultation Room	Patient Doctor
Waiting	Treatment Room	Patient
Waiting	Treatment Room	Patient Equipment
Waiting	Treatment Room	Patient Nurse

The nurses are in the nurses’ room unless they are called for a consultation or treatment. If there is no nurse available when a call is received, the next available nurse heads to the room which has sent the earliest request.

Equipment is held in the equipment room unless it is needed for treatment. After a treatment is performed, the equipment is sent to a cleaning room to be cleaned for next use.

A complete list of events in the system is given in the Table 5.1. This list is a key to transform location data into event data.

5.3.2. Description of the RTLS

The assumed tracking system is a generic real-time tracking system, which is similar to current applications in the industry. It is assumed to have perfect room level accuracy, that is, a tag can always be mapped to the correct room by means of its reported coordinates. Note that the actual location in a room will be rarely of interest in such a system. The tracking system collects location information at fixed intervals which are short enough to provide almost real-time positioning. Each reading is entered into a database table as a combination of tag ID, room ID and date/time. Furthermore, new readings for a tag are not entered into the database if the room ID has not changed since the previous reading. In other words, the database table holds only the readings for a tag when it is detected in a room for the first time. This helps keep the database relatively compact since there is no need to record subsequent readings of a tag at a location. It can safely be assumed that the tag is at a location until a new reading at a different location is observed.

It is assumed that doctors, nurses, mobile equipment and patients are tracked in the hospital unit. Furthermore, it is assumed that there are a finite number of tags available for patients and they are reusable while doctor, nurse and equipment tags are assigned to specific entities.

A significant feature of this model is the definition of doctors, nurses and equipments as entities. In previous healthcare simulation studies in the literature, such elements were generally modeled as resources or servers. It is customary to define a single entity, which would be the patient in this case, which is routed through the system receiving service. However for this thesis, since it is wanted to track the personnel and mobile equipment too, these must also be modeled as entities. This approach provides more detailed tracking data which can be mined to obtain more information on the behavior of the system.

The software agent queries this data to acquire service times for the processes to be monitored. This seamless integration makes it possible to test the developed software agent with different scenarios using the simulation model.

5.4. Data Parsing

The simulation model records the tracking data of the entities in a database structure. The primary table of interest includes the tag ID, location ID and timestamp for each reading. This format is a simplified example of the structure generally used in similar applications. Additional data tables can be utilized to map location IDs to actual location names, tag IDs to personnel names where each personnel is assigned a specific tag and tag IDs to patient names where it is possible.

The data acquired from a real-time tracking system can be used in different ways to derive performance parameters of the processes depending on how processes are defined in the system. A straightforward approach is to associate each location with an event type so that time spent by an entity at a location will be the time to complete that event. This approach has been used in similar studies in the literature [Amini, 2007]. Another candidate approach is to further breakdown the data exploiting the availability of tracking data of personnel and equipment. The events can be defined as combinations of entities at a location by employing prior knowledge of the workflow. In other words, an overlap of required entities for a process at a location indicates the related process. This approach was also proposed by Isken et al. (2004).

Let us illustrate the difference between two approaches using the proposed simulation model. A representation of the data parsing schemes is shown in Figure 5.8. According to the Approach 1, the time that a patient spends in the treatment room is considered as the treatment time. On the other hand, according to Approach 2, this time can be broken down into more descriptive events. For example, the time when a nurse and patient are in a treatment room can be considered as a waiting time since there is no equipment in the room. The time that a patient, a nurse and mobile equipment are in the treatment room can be taken as treatment set-up time. Thick lines represent the presence of each entity in a treatment room in a time window. A descriptive example of how these two approaches can be utilized using a tracking table is given in Appendix A.

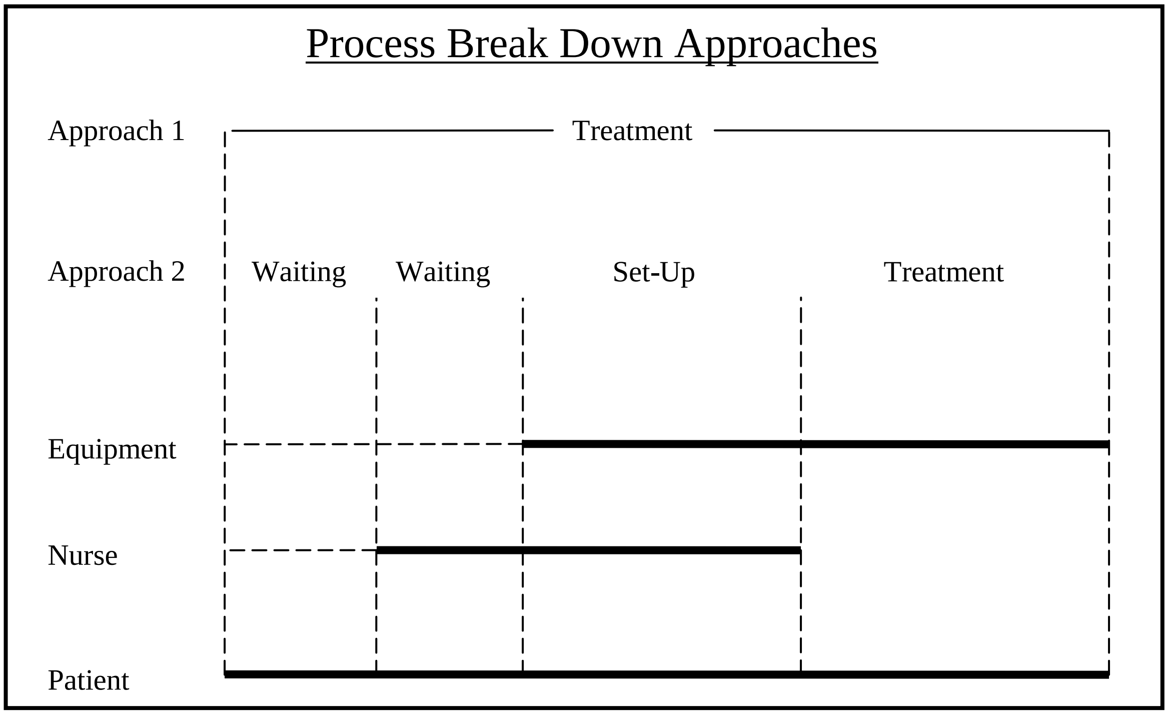


Figure 5.8 Representation of data parsing schemes

5.5. Data Cleaning

In any application of real time location systems, cleaning the tracking data is a crucial step. Inaccuracy in the tracking data can easily propagate into any analysis performed using this data. In this chapter we present an algorithm developed to clean real time tracking data.

Assume an RTLS scheme as described in 5.3.2 where each tag reports its position at fixed time intervals which are short enough to provide almost real time tracking. Each reading is recorded as a combination of tag ID, location ID and timestamp. In such a system, the reported location ID at each reading can be modeled to come from a discrete distribution given the actual location of the tag for an imperfect RTLS. This relaxes the “perfect room level accuracy” assumption and allows a more realistic modeling of the RTLS. Furthermore, assume that this discrete distribution, which shows the accuracy characteristics of the RTLS, is known. Although the algorithm is constructed with this assumption, it may not be true in a real world application. In a case where the accuracy characteristics of the RTLS are not precisely known, an iterative approach can be utilized. This approach is described in the following sections and experiments are performed in the Results & Discussions section.

A sliding window procedure is used to eliminate tracking errors in the stream of location IDs for a single tag. Let $L_1, L_2, \dots, L_k$ be the possible locations that the tag can be at. Note that probability P(reported location is L_i | actual location is L_j) is known for every combination of locations from the assumption that the accuracy characteristics of the RTLS is known. Let W be a window of consecutive observations (reported locations) for a specific tag, whose size is given as w, which is an odd number. For such a case,

$$P(L_i|W) = \frac{P(W|L_i)P(L_i)}{P(W)} = \frac{P(W|L_i)P(L_i)}{\sum_{j=1}^{k} P(W|L_j)P(L_j)} \tag{5.1}$$

is the probability of the tag being in location L_i given the window of reported locations. In the equation, $P(W|L_j)$ can be calculated using a multinomial distribution and $P(L_j)$ can be assigned from prior knowledge of the workflow for each location L_j. Comparison of calculated conditional probabilities gives the deduced location of the tag during the reading in the middle of the window. In other words, the result of each window calculation is assigned to the reading in the middle of that window.

5.5.1. Using Prior Knowledge of Workflow

Now assume that a Markov Chain is constructed using prior knowledge of the workflow where the states are the locations that a tag can be at during each reading. The Markov Chain is constructed such that the average length of stay at each state is equal to the average length of stay of the tag at each location in the system where stay lengths are expressed in number of reading intervals. The transition probabilities between states are also based on the transition probabilities of the tag between the locations. Ideally, this Markov Chain is the perfect representation of the workflow and its transition probabilities can be substituted for $P(L_j)$ for every j in Eq. 5.1, given the deduced location of the tag in the previous window. As an illustration to construction of the Markov Chain, let m_j be the average length of stay at a location L_j and p_{jj} be the probability that the tag is going to be at location L_j at time t+1 given that it was at the same location at time t. A simple conditional expression can be written as

$$m_j = p_{jj}\left(1 + m_j\right) + (1 - p_{jj}) \tag{5.2}$$

and it can be derived that

$$p_{jj} = (m_j - 1)/m_j \tag{5.3}$$

p_{ji}'s for $i = 1,2, \ldots, j-1, j+1, \ldots, k$ (transition probability into a different state) can be calculated by multiplying $(1 - p_{jj})$ with the probability of ending up in that state given that there is a transition.

While the algorithm is developed assuming that such a Markov Chain can be constructed, this may not be possible in a real world application. In such a case, an iterative method can be utilized, which is described in the following sections.

5.5.2. Without Using Prior Knowledge of Workflow

If there is no prior knowledge of the underlying system and no estimation can be done on the transition probabilities, Eq. 5.1 reduces to

$$P(L_i|W) = \frac{P(W|L_i)P(L_i)}{P(W)} = \frac{P(W|L_i)}{\sum_{j=1}^{k} P(W|L_j)} \tag{5.4}$$

because $P(L_1) = P(L_2) = \cdots = P(L_k)$. Performing data cleaning using this equation has the potential to produce satisfactory results under certain conditions, which generally lead to high performance in data cleaning using the proposed algorithm. More specifically, the algorithm performs better when the minimum length of the stay that is desired to be detected is long, the analysis window is large and the accuracy of the RTLS is high.

Even if no prior knowledge of the workflow is available, data can be cleaned more efficiently by performing a second pass on the already cleaned data with Eq. 5.4. The inadequacy in cleaning by the initial step results in very short stays which are a result of the randomness stemming from the inaccuracies in the RTLS and not actual visits to indicated locations. In fact, the algorithm is not supposed to detect stays which are shorter than half the size of the analysis window. Employing this idea, stays which are shorter than a determined threshold are ignored after the data is cleaned using Eq. 5.4.

5.5.3. Selection of Window Size

Selection of window size is one of the most critical parts of the algorithm. While a small window can be prone to lots of variation and provide inadequate data cleaning, a large window can easily miss short durations of stays at locations because of the large degree of aggregation imposed. Ideally, the window should be the smallest size which can provide the desired level of confidence for cleaning. In practice, different factors like accuracy of the RTLS and stay lengths should be considered while determining the window size. Monte Carlo simulation experiments can be set up to determine the best performing window size under certain conditions. Also upper and lower limits cans be set for the window size using analytical considerations. A lower limit can be set such that the probability of detecting the location accurately is larger than the desired confidence for a window of observations which are entirely from the same location. An upper limit can be set as two times the length of the shortest stay that is desired to be detected so that such a stay is not missed by the algorithm.

5.5.4. Imperfect Knowledge of Workflow and RTLS Accuracy

If the algorithm is preferred to be employed using prior knowledge of workflow but the transition probabilities cannot be estimated precisely, an iterative approach can be used with initial estimates. These initial estimates may be derived from preliminary studies of the system or expert judgment. Unless the initial estimates are very misleading, the algorithm provides relatively cleaner data than the raw form. After a preliminary round of cleaning is performed, the cleaned data can be used to derive new estimations of transition probabilities. This readjustment of transition probabilities are carried out until the difference between estimations of transition probabilities obtained from consecutive iterations is small enough.

The algorithm can also be employed by using initial estimates of the distributions of reported locations. If the accuracy characteristics of the RTLS are not precisely known, iterations can be performed in order to better estimate the distribution parameters. After a preliminary round of cleaning is performed, the cleaned data is assumed to be completely accurate and compared to raw data in order to derive distributions of reported locations given the true locations. The readjustment of estimations of distributions should be carried out until consistent estimations are obtained from consecutive iterations.

This iterative approach improves the applicability of the developed algorithm in situations where precise knowledge is not available about the RTLS and workflow. It also enables the algorithm to adapt to changes in the system. If the underlying system is subject to changes or the performance of the RTLS is not consistent, the probability distributions can be updated regularly using the iterative approach described above. This way, the algorithm can adjust itself to moderate changes in the system.

6. Results and Discussion

In this chapter, the results of the analyses performed are presented.

6.1. Effect of Late Detection on Parameter Estimation

Due to the randomness in the data it is never possible to be sure of the exactness of a detected change point even with manually created data. Depending on the coefficient of variation (ratio of standard deviation to mean) of the original distribution and the level of the change, the actual change point may be in the vicinity of the detected change point or it may be off by several observations.

For the software agent, the inaccuracy of a detected change point affects the analysis performed after detection. More specifically, a late detection causes the loss of data points after the actual change point but before the detected change point. For instance in the case of immediate learning after change point, this affects the number of data points used in fitting a learning model and may result in a bad fit and poor forecast of the steady state level.

A primary objective of fitting the learning curve is to be able to forecast the eventual steady state level of the process after learning has eased. A set of simulation experiments were performed to test the effect of late detection on the accuracy of forecast of steady state level. For each replication, a stream of 100 random numbers was generated such that first 50 of these data points come from a normal distribution. The 51st point in the stream was set as the change point and the first data point on the learning curve. The coefficient of variation, c_v, was kept constant for every generated data point. The distribution of the generated data is given in Eqs. 6.1 and 6.2.

$$y_x \sim N(a+b, c_v*(a+b)) \qquad x=1,2,\dots,50 \tag{6.1}$$

$$y_x \sim N(a+b*\exp[-c*(x-50)]\,, c_v*\{a+b*\exp[-c*(x-50)]\})$$

$$x=51,52,\dots,100 \tag{6.2}$$

Throughout the experiment a and b were set to 5. 1000 replications were run for different combinations of c_v and c, which are the coefficient of variation and the rate of learning respectively. Note that the steady state level of the learning equation is given by a.

For each replication, change point detection algorithm was run and the detected change point was recorded. Replications with false alarms (detected change points when there is no change in the process) were discarded in this experiment. Afterwards, the nonlinear fitting algorithm was run with the data stream after the detected change point as the input. Then it was run once more with the data stream after the actual change point as the input. Errors in the estimated steady state level were calculated for both cases in each replication. Following the rules of paired observation test, confidence intervals were calculated for the difference of errors. A portion of the calculation table is given below for illustration where the last column is the difference of errors. Note that the errors are given as fractions of the actual steady state level which is 5.

Table 6.1 Calculation table example of simulation runs

Replication	Detected change point	Estimated a via detected cp	Error1	Estimated a via actual cp	Error2	Error1 - Error2
84	53	7.4429	0.4886	6.9810	0.3962	0.0924
85	52	7.2198	0.4440	7.1814	0.4363	0.0077
86	52	6.9103	0.3821	6.8623	0.3725	0.0096
87	50	6.7462	0.3492	7.2519	0.4504	-0.1011
88	48	7.2652	0.4530	7.2652	0.4530	0.0000
89	51	7.5729	0.5146	7.5729	0.5146	0.0000
90	52	7.0787	0.4157	6.7356	0.3471	0.0686
91	48	6.5313	0.3063	6.7285	0.3457	-0.0394
92	53	7.3091	0.4618	7.2384	0.4477	0.0141
93	52	7.1881	0.4376	6.8900	0.3780	0.0596
94	51	6.9771	0.3954	6.9771	0.3954	0.0000
95	51	7.0917	0.4183	7.0917	0.4183	0.0000
96	51	7.3801	0.4760	7.3801	0.4760	0.0000

Summary of the results for 1000 replications are given in Table 6.2 for different combinations of c_v and c. Half widths of the confidence intervals are calculated for confidence level of 95%. A positive mean of differences of errors is an indicator that the estimate of the steady state level was adversely affected by the inaccuracy in change point detection.

Table 6.2 Summary of results for effects of late detection test

Difference of errors in estimated parameter a for detected and actual change points				
c_v	c:	0.01	0.03	0.05
0.01	Mean	0.0029	-0.0001	0.0045
	St. Dev.	0.0511	0.0214	0.0145
	Half Width	0.0032	0.0013	0.0009
0.05	Mean	0.0297	0.0070	0.0033
	St. Dev.	0.0617	0.0477	0.0224
	Half Width	0.0038	0.0030	0.0014
0.1	Mean	0.0211	0.0172	0.0080
	St. Dev.	0.1128	0.0807	0.0510
	Half Width	0.0070	0.0050	0.0032
0.2	Mean	N/A	0.0142	0.0092
	St. Dev.	N/A	0.1517	0.1126
	Half Width	N/A	0.0094	0.0070

The most important result that can be deduced from Table 6.2 is that the adverse effect of late detection on estimation of the learning model parameters is very minor. The largest difference recorded was 2.9% on average. For some of the cases the confidence interval includes 0 which means there is no statistical difference between the errors at 95% confidence level. For other cases, although the confidence interval does not include zero, average difference is small enough to be ignored from an application oriented point of view. It should be noted that the accuracy of estimation depends also on the total number of data points accumulated after the actual change point, which is kept constant in this case. “N/A” stands for the case where the statistics could not be reported since the change point could not be found by the detection algorithm within the 50 data points of the learning curve.

Another part of the results is given in Table 6.3. This table summarizes the accuracy of change point detection on average. Note that the actual change point is the 51st data point. Combining the results of Table 6.2 and Table 6.3 it can be seen that the adverse effect of late detection on the estimation of learning model steady state parameter is negligible even for the cases where the change point was detected 18 data points late on average. Also it can be observed that the accuracy of detection is directly proportional with the rate of learning and inversely proportional with coefficient of variation, which is intuitive.

Table 6.3 Accuracy of change point detection

Detected change points				
c_v	c:	0.01	0.03	0.05
0.01	Mean	51.6120	50.0420	49.7440
	St. Dev.	1.6930	1.3327	1.2410
	Half Width	0.1049	0.0826	0.0769
0.05	Mean	59.9150	53.2520	51.6850
	St. Dev.	4.1968	2.2328	1.7305
	Half Width	0.2601	0.1384	0.1073
0.1	Mean	68.2830	57.1700	54.2530
	St. Dev.	7.3966	3.4027	2.5705
	Half Width	0.4584	0.2109	0.1593
0.2	Mean	N/A	64.7200	59.5490
	St. Dev.	N/A	6.0956	4.3177
	Half Width	N/A	0.3778	0.2676

Table 6.4 Summary of results for effect of final steady state level test

Difference of errors in estimated parameter a for detected and actual change points								
SS Level:	1	2	3	4	5	6	7	8
Mean	0.0802	0.0434	0.0241	0.0222	0.0128	0.0105	0.0076	0.0023
St. Dev.	0.1091	0.0567	0.0332	0.0287	0.0176	0.0151	0.0113	0.0055
Half Width	0.0513	0.0302	0.0151	0.0157	0.0080	0.0059	0.0038	-0.0008
Detected change points								
Mean	53.6710	54.2970	54.9620	55.9370	57.1530	59.1960	62.2070	68.4220
St. Dev.	2.2751	2.4365	2.5556	3.0053	3.3872	3.9741	5.0918	8.0075
Half Width	0.1410	0.1510	0.1584	0.1863	0.2099	0.2463	0.3156	0.4963

Another set of simulation experiments was performed to test the effect of steady state level of the learning curve on accuracy of estimation. For this case c_v was set to 0.1 and c was set to 0.03. The in-control process was generated from $N(10,1)$. The experiment was conducted for different steady state levels of the learning curve. It can be deduced that when the ratio of the final steady state level to the initial steady state level is high, the detection is late but the error is negligible when compared to perfect detection case.

6.2. Evaluation of Data Cleaning Model

Monte Carlo simulation experiments are conducted in order to evaluate the effectiveness of the developed algorithm under different conditions. Synthetic tracking data is generated from a simple scenario of tag movement for each replication. Then this tracking data is cleaned utilizing the algorithm both with and without the prior knowledge of workflow using different window sizes. For each version of the algorithm, performance metrics like total number of transitions identified and length of stay at each location are calculated. Summary of results are given for each case considered.

According to the scenario, the system has three different locations with location IDs A, B, C. The tag moves in the system in the following fashion: It starts at location A and stays there for 400 seconds. Then it moves to location B to stay for 90 seconds. Finally it moves to location C to stay for 300 seconds. The location of the tag is reported to the server every 10 seconds with given accuracy characteristics.

Raw tracking data is cleaned using three different window sizes: 7, 9 and 11. For each window size, the algorithm is run both with and without prior knowledge. Four different secondary cleaning thresholds (3,4,5,6) are used for no prior knowledge cleaning. As an initial condition, all versions of the algorithm use the knowledge that the tag visits location A first. The algorithm cannot be run for some readings at the beginning and end of the data stream because there are not enough readings to form an analysis window. Therefore the analysis starts with the 6th reading in the data stream and ends at the 74th reading (there are 79 readings in the data stream). These boundary effects cause a minor error in determination of stay lengths at locations A and C by the algorithm. As a result of this, the algorithm can determine the length of stay at location A as 35 reading intervals and the length of stay at C as 25 reading intervals in the ideal case. (Each reading interval is 10 seconds).

Detailed example of data cleaning calculations using prior knowledge of the workflow is given in Appendix B. An example of data cleaning without using prior knowledge of the workflow is given in Appendix C.

6.2.1. Case 1: Base Scenario

In the base scenario, it is assumed that perfect knowledge of accuracy characteristics of the RTLS and workflow are available. The accuracy distribution of the RTLS is given in Table 6.5. According to the table, when the tag is actually in location A, the reported location is A with probability 0.8, B with probability 0.1 and C with probability 0.1.

Table 6.5 Accuracy distribution of the RTLS for Case 1

Actual Location	Reported location			
		A	B	C
	A	0.8	0.1	0.1
	B	0.1	0.8	0.1
	C	0.1	0.1	0.8

Table 6.6 Markov Chain representation of the workflow for Case1

$$\begin{Vmatrix} 39/40 & 1/40 & 0 \\ 0 & 8/9 & 1/9 \\ 0 & 0 & 30/30 \end{Vmatrix}$$

The Markov chain representation of the workflow is given in Table 6.6. Note how the average length of stay at every state is equal to the length of the stay at the associated location (expressed in number of reading intervals) in the given scenario. The transition probabilities between the locations are set such that the algorithm knows exactly which location should follow the previous one.

Table 6.7 Simulation results for Case 1

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	998	628	872	921	935	932
		<2	0	2	0	9	17	28	51
		>2	1000	0	372	119	62	37	17
	Average stay length	at A	3.72	35.87	28.74	32.94	33.83	34.28	34.72
		at B	1.91	8.24	8.08	8.85	8.88	8.85	8.74
		at C	3.09	24.89	22.09	24.42	24.86	25.04	25.13
9	Number of transitions	2	0	998	777	938	950	950	923
		<2	0	2	0	10	19	33	70
		>2	1000	0	223	52	31	17	7
	Average stay length	at A	3.72	35.96	32.03	33.99	34.34	34.58	34.91
		at B	1.91	8.29	8.46	8.90	8.88	8.85	8.67
		at C	3.09	24.76	23.24	24.93	25.09	25.16	25.24
11	Number of transitions	2	0	998	873	963	963	951	913
		<2	0	2	1	10	27	43	83
		>2	1000	0	126	27	10	6	4
	Average stay length	at A	3.72	36.06	33.19	34.33	34.60	34.71	34.94
		at B	1.91	8.27	8.66	8.96	8.93	8.87	8.66
		at C	3.09	24.67	24.32	25.16	25.27	25.29	25.31

The results of 1000 replications are summarized in Table 6.7. It can be seen that number of transitions were detected correctly in 998 replications out of 1000 for all three window sizes with prior knowledge of workflow. The stay lengths at locations were also estimated precisely for these cases. When no prior knowledge of workflow was used, window size 11 with a secondary cleaning threshold of 3 or 4 performed best in terms of number of transitions detected. Both cases also provided precise estimations of stay lengths. Higher thresholds of secondary cleaning resulted in undetected transitions in more replications because these transitions were treated as errors and eliminated after the initial cleaning step. Smaller window sizes resulted in inadequate cleaning which caused more reporting errors to be accepted as transitions.

In general, among the configurations considered, larger windows provided better cleaning if the threshold for secondary cleaning is chosen correctly. A too high threshold caused more transitions to be missed and a too low threshold caused more errors to be accepted as transitions. The results suggest that the number of replications, where the number of transitions was detected correctly, have a local maximum at a certain secondary cleaning threshold. Selection of a different threshold results in suboptimal cleaning performance.

6.2.2. Case 2: More flexible transition probabilities

In the base scenario, a perfect implementation of the algorithm was tested under specific conditions. The workflow was the same for all replications and the algorithm used this information for cleaning. However, in a real world application there may be variations in the workflow both systematic and nonsystematic. Therefore, the transition probabilities should be assigned more flexibly. In this scenario, the simulation experiment is repeated with an updated Markov Chain which allows transitions between all of the locations. The transition matrix is given in Table 6.8.

Table 6.8 Markov Chain representation of the workflow for Case 2

$$\begin{Vmatrix} 39/40 & 1/80 & 1/80 \\ 1/18 & 8/9 & 1/18 \\ 1/60 & 1/60 & 29/30 \end{Vmatrix}$$

The results of 1000 replications are given in Table 6.9. It can be seen that while the algorithm performed best with window size 11 using prior knowledge of workflow in terms of the number of transitions detected, performance without prior knowledge was also competitive when secondary cleaning was done. Estimation of average stay lengths was slightly better in the latter case. The performance of the algorithm was found to be satisfying for all window sizes with prior knowledge or with an appropriate choice of secondary cleaning threshold.

Table 6.9 Simulation results for Case 2

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	940	588	848	906	932	937
		<2	0	20	1	10	17	33	48
		>2	1000	40	411	142	77	35	15
	Average stay length	at A	3.67	36.33	28.17	32.71	33.73	34.42	34.78
		at B	1.92	8.27	8.10	8.81	8.89	8.85	8.80
		at C	3.13	23.57	21.75	24.16	24.65	24.99	25.11
9	Number of transitions	2	0	938	746	927	947	943	920
		<2	0	29	4	13	20	37	69
		>2	1000	33	250	60	33	20	11
	Average stay length	at A	3.67	36.43	31.79	33.93	34.33	34.59	34.89
		at B	1.92	8.21	8.47	8.87	8.88	8.84	8.68
		at C	3.13	23.69	22.90	24.87	25.05	25.13	25.20
11	Number of transitions	2	0	963	870	946	955	952	916
		<2	0	26	3	18	27	41	80
		>2	1000	11	127	36	18	7	4
	Average stay length	at A	3.67	36.69	33.25	34.22	34.53	34.70	34.95
		at B	1.92	8.24	8.61	8.89	8.90	8.85	8.66
		at C	3.13	23.84	24.30	25.10	25.21	25.30	25.30

6.2.3. Overestimation of RTLS performance

In a real world application, it may not be possible to exactly know the accuracy characteristics of the RTLS or the accuracy of the RTLS may decrease after the installation due to decreasing power of the batteries or other changes in the system. In such cases, RTLS performance may be overestimated. In this scenario, the experiment is repeated with the set up in Case 2 with one difference. The accuracy of the RTLS is overestimated using the values in Table 6.10. These distributions are used in Bayesian calculations while the readings are generated from the distributions which are given in Table 6.5.

Table 6.10 Assumed accuracy distributions of the RTLS for Case 3

		Reported location		
Actual Location		A	B	C
	A	0.9	0.05	0.05
	B	0.05	0.9	0.05
	C	0.05	0.05	0.9

Table 6.11 Simulation results for Case 3

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	881	603	868	909	940	940
		<2	0	12	1	6	15	23	44
		>2	1000	107	396	126	76	37	16
	Average stay length	at A	3.68	34.83	28.53	32.89	33.67	34.26	34.70
		at B	1.92	7.99	8.13	8.93	8.98	8.97	8.88
		at C	3.12	23.88	21.68	24.15	24.55	24.89	25.04
9	Number of transitions	2	0	946	777	933	945	939	917
		<2	0	10	5	12	22	41	73
		>2	1000	44	218	55	33	20	10
	Average stay length	at A	3.68	35.56	32.14	34.04	34.35	34.68	34.97
		at B	1.92	8.14	8.55	8.90	8.90	8.80	8.63
		at C	3.12	24.38	23.01	24.83	24.99	25.09	25.19
11	Number of transitions	2	0	972	854	962	959	943	913
		<2	0	12	3	13	26	50	82
		>2	1000	16	143	25	15	7	5
	Average stay length	at A	3.68	35.83	33.01	34.37	34.53	34.79	34.97
		at B	1.92	8.21	8.67	9.01	8.99	8.90	8.72
		at C	3.12	24.61	24.02	25.06	25.12	25.17	25.19

The results of 1000 replications are summarized in Table 6.11. It can be seen that the only configuration of the algorithm that was significantly affected by the overestimation was using the prior knowledge of workflow with a window size of 7. Larger window sizes and configurations that do not use prior knowledge were robust to the overestimation of RTLS accuracy.

6.2.4. Case 4: Underestimation of RTLS performance

In this scenario the experiment is repeated by using the set up in Case 2 with one alteration. The RTLS accuracy distributions given in Table 6.12 are used for Bayesian calculations instead of the correct distributions. The distributions in Table 6.12 underestimate the accuracy of the RTLS.

Table 6.12 Assumed accuracy distributions of the RTLS for Case 4

Actual Location	Reported location			
		A	B	C
	A	0.7	0.15	0.15
	B	0.15	0.7	0.15
	C	0.15	0.15	0.7

Simulation results are summarized in Table 6.13. It can be deduced that no configuration of the algorithm was significantly affected from underestimation of RTLS accuracy.

Table 6.13 Simulation results for Case 4

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	940	611	856	903	933	933
		<2	0	35	2	9	23	35	54
		>2	1000	25	387	135	74	32	13
	Average stay length	at A	3.62	36.61	28.59	32.85	33.79	34.47	34.90
		at B	1.88	8.24	8.00	8.79	8.81	8.79	8.73
		at C	3.07	23.64	21.90	24.20	24.70	25.00	25.09
9	Number of transitions	2	0	944	765	929	936	936	914
		<2	0	34	6	16	27	43	75
		>2	1000	22	229	55	37	21	11
	Average stay length	at A	3.62	36.65	32.01	34.14	34.46	34.74	35.02
		at B	1.88	8.21	8.39	8.76	8.76	8.72	8.56
		at C	3.07	23.75	23.05	24.88	24.98	25.13	25.17
11	Number of transitions	2	0	954	849	945	947	939	905
		<2	0	39	6	21	33	48	88
		>2	1000	7	145	34	20	13	7
	Average stay length	at A	3.62	36.83	33.00	34.44	34.67	34.78	35.03
		at B	1.88	8.17	8.52	8.86	8.85	8.78	8.58
		at C	3.07	23.88	24.21	25.03	25.12	25.18	25.21

6.3.5. Case 5: Iterative Methods

When an RTLS is being set up in an environment, no statistical information about the RTLS or the underlying system may be available. In such a case it may be necessary to estimate the distribution parameters to be used in data cleaning completely by judgment. The algorithm developed makes it possible to iteratively generate better estimates of RTLS accuracy parameters and transition probabilities. To achieve this, a preliminary cleaning is performed. Afterwards, adjusted estimates can be derived from the cleaned data. Adjustment of parameters should be carried on until the difference between the estimates from consecutive iterations is small enough.

In this scenario, a preliminary cleaning is performed on 100 replications using the RTLS accuracy distributions given in Table 6.14 and transition probabilities given in Table 6.15 while the data is generated using the distributions given in the base scenario.

Table 6.14 Assumed RTLS accuracy in iteration 1 for Case 5

Actual Location		Reported location		
		A	B	C
	A	0.5	0.25	0.25
	B	0.25	0.5	0.25
	C	0.25	0.25	0.5

Table 6.15 Assumed transition probabilities in iteration 1 for Case 5

$$\begin{Vmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{Vmatrix}$$

Table 6.14 implies an arbitrary assumption of 50% RTLS accuracy and Table 6.15 implies no knowledge of the workflow. The results of 100 replications are summarized in Table 6.16.

The algorithm provided satisfactory cleaning without using prior knowledge of the workflow even with a very bad estimate of RTLS accuracy when secondary cleaning was utilized. Using prior knowledge did not offer any additional cleaning over initial cleaning with no prior knowledge (and no secondary cleaning). This result is intuitive since the transition probabilities used in cleaning do not provide any information on the movement of the tag. However, average stay length estimations were reasonably precise for both cases. Table 6.17 summarizes estimations calculated from the preliminary cleaning. These values are derived by assuming that the cleaned data is completely accurate. RTLS accuracy column shows the fraction of readings where the raw data and the cleaned data are the same. Fractions of transitions give a breakdown of transitions given the initial location. For example, when window size of 7 was used, 79% of the transitions from location A were to location B. Note that the fraction of transitions from a location sum up to 1.

Table 6.16 Simulation results for Case 5 after the first iteration

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	54	54	80	89	91	90
		<2	0	1	1	2	2	4	7
		>2	100	45	45	18	9	5	3
	Average stay length	at A	3.61	28.08	28.08	32.43	33.99	34.29	34.70
		at B	1.82	7.69	7.69	8.74	8.90	8.85	8.70
		at C	3.08	21.49	21.49	23.41	24.09	24.62	24.72
9	Number of transitions	2	0	70	70	93	93	93	88
		<2	0	2	2	2	3	5	10
		>2	100	28	28	5	4	2	2
	Average stay length	at A	3.61	31.15	31.15	34.45	34.48	34.80	35.05
		at B	1.82	8.39	8.39	8.75	8.75	8.68	8.43
		at C	3.08	22.14	22.14	24.71	24.85	25.06	25.06
11	Number of transitions	2	0	83	83	95	94	91	89
		<2	0	2	2	3	4	7	11
		>2	100	15	15	2	2	2	0
	Average stay length	at A	3.61	33.23	33.23	34.65	34.68	34.84	35.42
		at B	1.82	8.44	8.44	8.80	8.77	8.63	8.43
		at C	3.08	24.06	24.06	25.06	25.06	25.06	25.15

Table 6.17 Estimations from the first iteration

Window Size	RTLS Accuracy	Fractions of transitions					
		A->B	A->C	B->A	B->C	C->A	C->B
7	0.790	0.790	0.210	0.224	0.776	0.389	0.611
9	0.784	0.776	0.224	0.081	0.919	0.533	0.467
11	0.781	0.927	0.073	0.065	0.935	0.400	0.600

Window size 11 is selected to be used in this approach since it provided the best cleaning for the scenarios that were previously examined. Realize that the assumed RTLS accuracy and the average stay lengths are already very close to true values while the fractions of transitions require one more iteration. The RTLS accuracy distributions and transition probabilities used in the next 100 replications are given in Table 6.18 and Table 6.19. Accuracy distributions are assigned simply by using the RTLS accuracy derived from the first run. Transition probability matrix is constructed such that the average stay length at a state is equal to the average stay length estimated from the first iteration. Transition rates into states are adjusted according to the fractions of transitions estimated. Detailed explanation of derivation of transition probabilities and accuracy distributions are given in Appendix D.

Table 6.18 Assumed RTLS accuracy in iteration 2 for Case 5

Actual Location	Reported location			
		A	B	C
	A	0.78	0.11	0.11
	B	0.11	0.78	0.11
	C	0.11	0.11	0.78

Table 6.19 Assumed transition probabilities in iteration 2 for Case 5

$$\begin{Vmatrix} 32/33 & 0.93/33 & 0.07/33 \\ 0.06/8 & 7/8 & 0.94/8 \\ 0.4/24 & 0.6/24 & 23/24 \end{Vmatrix}$$

The results of the second iteration are given in Table 6.20 and Table 6.21. The results show that the cleaning performed was very successful in terms of all the statistics derived. The algorithm provided very good estimation of stay lengths, number of transitions, RTLS accuracy and fraction of transitions. Note that no fraction of transitions could be calculated for transitions from C with window size 11 since there were no transitions detected from C in any of 100 replications. It can be deduced that the algorithm managed to estimate the necessary parameters for cleaning very precisely after 100 runs. It is noteworthy to point out that the estimated parameters are very close the parameters that should be assigned for ideal performance as in Case 1.

Table 6.20 Simulation results for Case 5 after the second iteration

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	96	57	83	94	98	94
		<2	0	0	0	1	1	1	5
		>2	100	4	43	16	5	1	1
	Average stay length	at A	3.69	35.70	27.54	32.33	33.90	34.52	34.72
		at B	1.98	9.02	8.08	8.95	9.05	9.13	8.93
		at C	2.91	23.51	20.85	23.92	24.84	25.14	25.14
9	Number of transitions	2	0	98	73	94	96	95	93
		<2	0	1	0	1	3	4	7
		>2	100	1	27	5	1	1	0
	Average stay length	at A	3.69	36.00	31.10	33.83	34.51	34.55	34.92
		at B	1.98	9.06	8.75	9.21	9.16	9.12	8.95
		at C	2.91	23.77	22.35	24.80	25.13	25.13	25.13
11	Number of transitions	2	0	100	88	98	98	96	92
		<2	0	0	0	1	2	4	8
		>2	100	0	12	1	0	0	0
	Average stay length	at A	3.69	35.83	32.84	34.49	34.55	34.63	34.83
		at B	1.98	9.34	9.00	9.27	9.24	9.16	8.96
		at C	2.91	23.83	24.23	25.10	25.21	25.21	25.21

Table 6.21 Estimations from the second iteration

Window Size	RTLS Accuracy	Fractions of transitions					
		A->B	A->C	B->A	B->C	C->A	C->B
7	0.780	1.000	0.000	0.019	0.981	0.000	1.000
9	0.780	0.990	0.010	0.000	1.000	0.000	1.000
11	0.777	1.000	0.000	0.000	1.000	N/A	N/A

7. Conclusion

We have developed a software agent that makes use of RTLS data to carry out real time performance monitoring and analysis. Once connected to a data source the developed agent monitors the performance metric of a repeatedly executed task. It is assumed that the performance metric to be monitored is attributable to a stay at a location whose length can be mined from the tracking data. It can identify steady state behavior in the performance, derive the distribution parameters of the performance metric, detect changes, model learning periods and forecast the future steady state level of performance. Several statistical methods have been employed and customized for the agent to perform these tasks. It can perform these tasks in real time, providing the human operator current information about the process monitored. The agent can be integrated into any RTLS system and work on the tracking data stored in a database. We have developed the simulation model of a hypothetical hospital unit to generate synthetic tracking data. This model is linked to the developed software agent to complete the framework.

We have conducted experiments to investigate the effect of late change point detection on the accuracy of the forecast of the future steady state level. These experiments showed that late detection of a change point does not have a significant effect on the forecast of the future steady state level of performance. We have also investigated the effect of coefficient of variation and the rate of learning on change point detection accuracy. The results showed that the accuracy of change point detection is directly proportional to the learning rate after a steady state region and inversely proportional to the coefficient of variation.

Inaccuracies which are inherent in any RTLS can make performance analysis using the tracking data impossible. We have developed a data cleaning algorithm that utilizes an assumed accuracy distribution of the RTLS. The algorithm can perform with or without using the knowledge of the workflow. It has been shown that accuracy characteristics of the RTLS and the transition probabilities between locations can be estimated via a preliminary cleaning run if they are not available. Monte Carlo simulation experiments showed that the algorithm is robust to estimation errors in RTLS accuracy and transition probabilities. This robustness of the algorithm along with the ability to estimate the necessary parameters makes the algorithm very suitable for practical applications. The algorithm could determine the number of transactions correctly in at least 95% of the replications for all of the cases considered when a “good” window size is selected. The performance was much higher under ideal or close to ideal conditions. Stay lengths at locations mined from the cleaned data were also very accurate. These results are found to be promising in terms of using the tracking data for performance monitoring and other purposes even if the RTLS has accuracy issues.

We have also built a simulation model of a hospital unit with an RTLS and suggested data parsing methods that can be used in such scenarios.

All of the components of the thesis are joined together to form a new approach for using real time tracking data. This study can be seen as an example of how tracking data can be processed and used in other applications. The data cleaning algorithm developed can be employed as a tool to clean tracking data for purposes other than the one proposed in this thesis.

References

Amini, M., R. F. Otondo, B. D. Janz and M. G. Pitts, 2007, "Simulation modeling and analysis: A collateral application and exposition of RFID technology", *Production and Operations Management*, Vol. 16, No. 5, pp. 586-598.

Baloff, N., 1971, "Extension of the Learning Curve--Some Empirical Results", *Operational Research Quarterly*, Vol. 22, No. 4, pp. 329-340.

Banks, J., J. S. Carson, B. L. Nelson and D. M. Nicol, 1984 *Discrete event system simulation*, Prentice Hall,Englewood Cliffs, NJ.

Brockwell, P. J. and R. A. Davis, 2002, *Introduction to time series and forecasting,* Springer.

Danford, D. A., J. D. Kugler, B. Deal, C. Case, R. A. Friedman, J. P. Saul, M. J. Silka and G. F. Van Hare, 1995, "The learning curve for radiofrequency ablation of tachyarrhythmias in pediatric patients. Participating members of the Pediatric Electrophysiology Society", Vol. 75, No. 8, p. 587.

Elnahrawy, E. and Nath, B., 2003, “Cleaning and Querying Noisy Sensors”, *Proceedings of the MobiCom's Second ACM International Workshop on Wireless Sensor Networks*, San Diego, USA.

Fahmy, H. M. and E. A. Elsayed, 2006, "Detection of linear trends in process mean", *International Journal of Production Research*, Vol. 44, No. 3, pp. 487-504.

Hudson, D. J., 1966, “Fitting Segmented Curves Whose Join Points Have to be Estimated”, *Journal of the American Statistical Association*, Vol. 61, No. 316, pp. 1097-1129

Isken, M. W., V. Sugumaran, T. J. Ward, D. Minds and W. Ferris, 2005, "Collection and preparation of sensor network data to support modeling and analysis of outpatient clinics", *Health Care Management Science*, Vol. 8, No. 2, pp. 87-99.

Jeffery, S. R., Garofalakis, M., and Franklin, M. J., 2006, “Adaptive cleaning for RFID data streams”, *Proceedings of the 32nd international Conference on Very Large Data Bases* (Seoul, Korea, September 12 - 15, 2006). U. Dayal, K. Whang, D. Lomet, G. Alonso, G. Lohman, M. Kersten, S. K. Cha, and Y. Kim, Eds. Very Large Data Bases. VLDB Endowment, 163-174.

Langley, G.L., K.M. Nolan, T.W. Nolan, C.L. Norman, and L.P. Proust, The Improvement Guide, San Francisco: Jossey-Bass Publishers, 1996.

Marquardt, D. W., 1963, "An algorithm for least-squares estimation of nonlinear parameters", Vol. 11, No. 2, pp. 431-441.

Miller, M. J., D. M. Ferrin, T. Flynn, M. Ashby, K. P. White Jr and M. G. Mauer, 2006, "Using RFID technologies to capture simulation data in a hospital emergency department", *Proceedings of 38th Conference on Winter Simulation* pp. 1365-1370, Winter Simulation Conference.

Ramakrishnan, S., K. Nagarkar, M. DeGennaro, M. Srihari, A. K. Courtney and F. Emick, 2004, "A study of the CT scan area of a healthcare provider", Vol. 2, pp. 2025-2031 .

Ramsay, C. R., A. M. Grant, S. A. Wallace, P. H. Garthwaite, A. F. Monk and I. T. Russell, 2001, "Assessment of the learning curve in health technologies", Vol. 16, No. 04, pp. 1095-1108.

Rossetti, M. D., G. F. Trzcinski and S. A. Syverud, 1999, "Emergency department simulation and determination of optimal attending physician staffing schedules", *Proceedings of 1999 Winter Simulation Conference*.

Walpole, R. E., R. H. Myers and S. L. Myers, 1989, *Probability and statistics for scientists and engineers*, New York: Macmillan.

Weng, M. L. and A. A. Houshmand, 1999, "Healthcare simulation: a case study at a local clinic", *Proceedings of 31st Conference on Winter Simulation: Simulation - A Bridge To the Future*, Vol. 2, pp. 1577-1584, ACM New York, NY, USA.

Appendix A: Data Parsing Approaches

Examples to data parsing approaches which are introduced in Chapter 5.4 are given below.

Approach 1 uses the tracking data of a single entity which receives service from the system (a patient in this case). Approach 2 uses the tracking data of all the entities that are involved in an operation. For instance, a patient, a nurse and a doctor have to be present in a consultation room for a consultation to take place.

Table A.1 shows a portion of raw tracking data. The data given has no tracking errors and it is simplified by keeping only the records that mark the entrance times to locations. Assume that consultation times are desired to be calculated from the data. Table A.2 summarizes the rows of Table A.1 that are used in mining of consultation times using Approach 1 and the resulting consultation times are given. Table A.3 summarizes the rows of Table A.1 that are used in mining of consultation times using Approach 2 and the resulting consultation times are given. In this approach, arrival time of the last entity that is involved in consultation marks the start of the event.

Table A.1 Raw data ready for data parsing

Record ID	**Entity**	**Location**	**Time Stamp**
211943	Nurse 2	Corridor	5/20/2009 15:28
211944	Doctor 2	Corridor	5/20/2009 15:28
211945	Patient 44	Register	5/20/2009 15:29
211946	Equipment 1	Equipment Cleaning	5/20/2009 15:29
211947	Nurse 3	Treatment Room 2	5/20/2009 15:29
211948	Patient 33	Waiting Room	5/20/2009 15:29
211949	Patient 35	Consultation Room 1	5/20/2009 15:29

211950	Patient 45	Register	5/20/2009 15:30
211951	Patient 27	Corridor	5/20/2009 15:30
211952	Patient 34	Corridor	5/20/2009 15:30
211953	Nurse 6	Corridor	5/20/2009 15:30
211954	Doctor 1	Corridor	5/20/2009 15:30
211955	Equipment 6	Treatment Room 3	5/20/2009 15:30
211956	Patient 34	Consultation Room 2	5/20/2009 15:31
211957	Patient 44	Corridor	5/20/2009 15:31
211958	Nurse 2	Consultation Room 1	5/20/2009 15:31
211959	Patient 27	Waiting Room	5/20/2009 15:31
211960	Doctor 2	Consultation Room 1	5/20/2009 15:32
211961	Patient 44	Waiting Room	5/20/2009 15:32
211962	Patient 45	Corridor	5/20/2009 15:32
211963	Equipment 9	Corridor	5/20/2009 15:32
211964	Nurse 6	Consultation Room 2	5/20/2009 15:33
211965	Doctor 1	Consultation Room 2	5/20/2009 15:33
211966	Patient 45	Waiting Room	5/20/2009 15:33
211967	Equipment 10	Corridor	5/20/2009 15:36
211968	Equipment 9	Treatment Room 1	5/20/2009 15:40
211969	Nurse 1	Corridor	5/20/2009 15:41
211970	Patient 35	Corridor	5/20/2009 15:41
211971	Patient 36	Corridor	5/20/2009 15:41
211972	Nurse 2	Corridor	5/20/2009 15:41
211973	Doctor 2	Corridor	5/20/2009 15:41
211974	Patient 35	Register	5/20/2009 15:42
211975	Patient 36	Consultation Room 1	5/20/2009 15:42
211976	Patient 34	Corridor	5/20/2009 15:42
211977	Patient 37	Corridor	5/20/2009 15:42
211978	Nurse 6	Corridor	5/20/2009 15:42
211979	Doctor 1	Corridor	5/20/2009 15:42
211980	Equipment 10	Treatment Room 4	5/20/2009 15:43
211981	Patient 34	Waiting Room	5/20/2009 15:43
211982	Patient 37	Consultation Room 2	5/20/2009 15:43
211983	Nurse 6	Nurses Room	5/20/2009 15:43
211984	Nurse 2	Consultation Room 1	5/20/2009 15:44
211985	Nurse 1	Consultation Room 2	5/20/2009 15:44

211986	Doctor 2	Consultation Room 1	5/20/2009 15:44
211987	Doctor 1	Consultation Room 2	5/20/2009 15:45
211988	Patient 2	Corridor	5/20/2009 15:46
211989	Patient 30	Corridor	5/20/2009 15:46
211990	Equipment 2	Corridor	5/20/2009 15:46
211991	Nurse 6	Corridor	5/20/2009 15:46
211992	Patient 30	Treatment Room 5	5/20/2009 15:47
211993	Patient 2	Register	5/20/2009 15:48
211994	Nurse 5	Corridor	5/20/2009 15:49
211995	Equipment 2	Equipment Cleaning	5/20/2009 15:49
211996	Nurse 6	Treatment Room 5	5/20/2009 15:49
211997	Nurse 5	Nurses Room	5/20/2009 15:51
211998	Nurse 4	Corridor	5/20/2009 15:53
211999	Patient 15	Corridor	5/20/2009 15:54
212000	Patient 29	Corridor	5/20/2009 15:54
212001	Equipment 5	Corridor	5/20/2009 15:54
212002	Nurse 5	Corridor	5/20/2009 15:54
212003	Patient 36	Corridor	5/20/2009 15:54
212004	Patient 38	Corridor	5/20/2009 15:54
212005	Nurse 2	Corridor	5/20/2009 15:54
212006	Doctor 2	Corridor	5/20/2009 15:54
212007	Patient 29	Treatment Room 6	5/20/2009 15:55
212008	Patient 36	Waiting Room	5/20/2009 15:55
212009	Patient 38	Consultation Room 1	5/20/2009 15:55
212010	Nurse 2	Nurses Room	5/20/2009 15:55
212011	Patient 15	Register	5/20/2009 15:56
212012	Patient 2	Corridor	5/20/2009 15:57
212013	Patient 37	Corridor	5/20/2009 15:57
212014	Patient 39	Corridor	5/20/2009 15:57
212015	Nurse 1	Corridor	5/20/2009 15:57
212016	Nurse 2	Corridor	5/20/2009 15:57
212017	Nurse 4	Consultation Room 1	5/20/2009 15:57
212018	Doctor 1	Corridor	5/20/2009 15:57

Table A.2 Consultation times using Approach 1

Arrival of the entity				Departure of the Entity				Time
Record ID	Entitiy	Location	Time Stamp	Record ID	Entity	Location	Time Stamp	mins
211949	Patient 35	Consultation Room 1	5/20/2009 15:29	211970	Patient 35	Corridor	5/20/2009 15:41	12
211956	Patient 34	Consultation Room 2	5/20/2009 15:31	211976	Patient 34	Corridor	5/20/2009 15:42	11
211975	Patient 36	Consultation Room 1	5/20/2009 15:42	212003	Patient 36	Corridor	5/20/2009 15:54	12
211982	Patient 37	Consultation Room 2	5/20/2009 15:43	212013	Patient 37	Corridor	5/20/2009 15:57	14

Table A.3 Consultation Times using Approach 2

Arrival of the entity				Departure of the Entity				Time
Record ID	Entity	Location	Time Stamp	Record ID	Entity	Location	Time Stamp	mins
211949	Patient 35	Consultation Room 1	5/20/2009 15:29	211970	Patient 35	Corridor	5/20/2009 15:41	9
211958	Nurse 2	Consultation Room 1	5/20/2009 15:31	211972	Nurse 2	Corridor	5/20/2009 15:41	
211960	Doctor 2	Consultation Room 1	5/20/2009 15:32	211973	Doctor 2	Corridor	5/20/2009 15:41	
211956	Patient 34	Consultation Room 2	5/20/2009 15:31	211976	Patient 34	Corridor	5/20/2009 15:42	9
211964	Nurse 6	Consultation Room 2	5/20/2009 15:33	211978	Nurse 6	Corridor	5/20/2009 15:42	
211965	Doctor 1	Consultation Room 2	5/20/2009 15:33	211979	Doctor 1	Corridor	5/20/2009 15:42	
211975	Patient 36	Consultation Room 1	5/20/2009 15:42	212003	Patient 36	Corridor	5/20/2009 15:54	10
211984	Nurse 2	Consultation Room 1	5/20/2009 15:44	212005	Nurse 2	Corridor	5/20/2009 15:54	
211986	Doctor 2	Consultation Room 1	5/20/2009 15:44	212006	Doctor 2	Corridor	5/20/2009 15:54	
211982	Patient 37	Consultation Room 2	5/20/2009 15:43	212013	Patient 37	Corridor	5/20/2009 15:57	12
211985	Nurse 1	Consultation Room 2	5/20/2009 15:44	212015	Nurse 1	Corridor	5/20/2009 15:57	
211987	Doctor 1	Consultation Room 2	5/20/2009 15:45	212018	Doctor 1	Corridor	5/20/2009 15:57	

Appendix B: Data Cleaning Calculations

Sample calculations for the data cleaning algorithm are given below.

Assume a system with three locations, namely A, B and C. Assume that the location of the tag is determined as location A in the previous calculation. Also assume that the cleaning window size is chosen as 9 and the window of observations are as given in Table B.1.

Table B.1 A window of observations

Index:	11	12	13	14	15	16	17	18	19
Observation:	A	A	A	B	A	B	C	A	A

The cleaning algorithm is applied using the given accuracy distributions in Table B.2 and transition probabilities in Table B.3.

Table B.2 Assumed accuracy distributions of the RTLS

Actual Location	Reported location			
		A	B	C
	A	0.8	0.1	0.1
	B	0.1	0.8	0.1
	C	0.1	0.1	0.8

Table B.3 Assumed transition matrix

$$\begin{Vmatrix} 39/40 & 1/80 & 1/80 \\ 1/18 & 8/9 & 1/18 \\ 1/60 & 1/60 & 1/30 \end{Vmatrix}$$

P(W|A) is the probability that the observations (reported locations) in W are obtained given that the tag is actually at location A. It can be calculated using a multinomial distribution.

$$P(W|A) = \frac{9!}{6! * 2! * 1!} * 0.8^6 * 0.1^2 * 0.1^1 = 0.066$$

$$P(W|B) = \frac{9!}{6! * 2! * 1!} * 0.1^6 * 0.8^2 * 0.1^1 = 1.613E - 05$$

$$P(W|C) = \frac{9!}{6! * 2! * 1!} * 0.1^6 * 0.1^2 * 0.8^1 = 2.016E - 06$$

Using the Bayesian relation, probability that the tag is actually at location A given the window of reported locations, P(A|W), can be calculated as given in the following equation.

$$P(A|W) = \frac{P(W|A)P(A)}{P(W|A)P(A) + P(W|B)P(B) + P(W|C)P(C)}$$

where P(A) is the probability that the tag is actually at location A. From the transition probabilities, this probability is equal to 39/40 given that the previous location was found to be A in the previous calculation by assumption. P(B) = 1/80 and P(C) = 1/80.

$$P(A|W) = \frac{0.066 * 0.975}{0.066 * 0.975 + 1.613 * 10^{-5} * 0.0125 + 2.016 * 10^{-6} * 0.0125} = 0.999$$

$$P(B|W) = \frac{1.613 * 10^{-5} * 0.0125}{0.066 * 0.975 + 1.613 * 10^{-5} * 0.0125 + 2.016 * 10^{-6} * 0.0125} = 3.131E - 06$$

$$P(C|W) = \frac{2.016 * 10^{-6} * 0.0125}{0.066 * 0.975 + 1.613 * 10^{-5} * 0.0125 + 2.016 * 10^{-6} * 0.0125} = 3.916E - 07$$

Note that $P(A|W) + P(B|W) + P(C|W) = 1$ as it should. Given the conditional probabilities, the algorithm decides that the true location of the tag at the 15th reading (the reading at the center of the window) is A.

Appendix C: Data Cleaning Without Prior Knowledge

An example to data cleaning without using prior knowledge is given below.

When no prior knowledge of the workflow is available, the Bayesian relation which is given as

$$P(A|W) = \frac{P(W|A)P(A)}{P(W|A)P(A) + P(W|B)P(B) + P(W|C)P(C)}$$

reduces to

$$P(A|W) = \frac{P(W|A)}{P(W|A) + P(W|B) + P(W|C)}$$

where A, B and C are the possible locations that a tag can be at. This is due to the fact that P(A), P(B) and P(C) can be taken as equal since the tag can be at any location at any time. In such a case, the data cleaning algorithm compares P(W|A), P(W|B) and P(W|C) to determine the true location of the tag. This constitutes the first step of cleaning (primary cleaning). Then the cleaned data stream is filtered again by discarding consecutive readings from a location if the number of consecutive readings is smaller than a threshold. This constitutes the secondary cleaning. Table C.1 illustrates secondary cleaning. The secondary cleaning threshold is chosen as 4. Realize how readings from C at indexes 14, 15 and 16 are discarded since the number of consecutive readings from C is 3. The readings from B at indexes 23, 24, 25 and 26 are accepted as a transition to location B.

Table C.1 . Secondary cleaning example

Reading Index	Primary Cleaning	Secondary Cleaning
1	A	A
2	A	A
3	A	A
4	A	A
5	A	A
6	C	A
7	A	A
8	A	A
9	A	A
10	B	A
11	B	A
12	A	A
13	A	A
14	C	A
15	C	A
16	C	A
17	A	A
18	A	A
19	A	A
20	A	A
21	A	A
22	A	A
23	B	B
24	B	B
25	B	B
26	B	B
27	A	B
28	B	B
29	C	C
30	C	C
31	C	C
32	C	C
33	C	C

Appendix D: Deriving Parameters for Data Cleaning

An example to calculation of transition probabilities is given below. This calculation can be carried out using the results of a preliminary cleaning or historical data collected via a reliable method. Summary of statistics derived after a preliminary cleaning run is given in Table D.1 and Table D.2.

Table D.1 Part 1 of preliminary cleaning results

Window size	Performance Metrics		Raw data	w/ Prior	w/o Prior	w/o Prior 2nd pass (3)	w/o Prior 2nd pass (4)	w/o Prior 2nd pass (5)	w/o Prior 2nd pass (6)
7	Number of transitions	2	0	54	54	80	89	91	90
		<2	0	1	1	2	2	4	7
		>2	100	45	45	18	9	5	3
	Average stay length	at A	3.61	28.08	28.08	32.43	33.99	34.29	34.70
		at B	1.82	7.69	7.69	8.74	8.90	8.85	8.70
		at C	3.08	21.49	21.49	23.41	24.09	24.62	24.72
9	Number of transitions	2	0	70	70	93	93	93	88
		<2	0	2	2	2	3	5	10
		>2	100	28	28	5	4	2	2
	Average stay length	at A	3.61	31.15	31.15	34.45	34.48	34.80	35.05
		at B	1.82	8.39	8.39	8.75	8.75	8.68	8.43
		at C	3.08	22.14	22.14	24.71	24.85	25.06	25.06
11	Number of transitions	2	0	83	83	95	94	91	89
		<2	0	2	2	3	4	7	11
		>2	100	15	15	2	2	2	0
	Average stay length	at A	3.61	33.23	33.23	34.65	34.68	34.84	35.42
		at B	1.82	8.44	8.44	8.80	8.77	8.63	8.43
		at C	3.08	24.06	24.06	25.06	25.06	25.06	25.15

Table D.2 Part 2 of preliminary cleaning results

Window Size	RTLS Accuracy	Fractions of transitions					
		A->B	A->C	B->A	B->C	C->A	C->B
7	0.790	0.790	0.210	0.224	0.776	0.389	0.611
9	0.784	0.776	0.224	0.081	0.919	0.533	0.467
11	0.781	0.927	0.073	0.065	0.935	0.400	0.600

The results from the preliminary cleaning using window size 11 (shaded cells) are employed for calculations. The average length of stay at location A was found to be 33.23 reading intervals which can be rounded down to 33. Using Eq. 5.3, transition probability from A to A is calculated as 32/33. From Table D.3 it can be seen that out of transition from A to other locations, approximately 93% were to location B. So the transition probability from A to B is obtained as 0.93/33. Similarly, transition probability from A to C is obtained as 0.07/33. The rest of the transition probabilities are calculated in the same manner and the transition matrix given in Table D.3 is obtained. This transition matrix is to be used in the next iteration of data cleaning.

Table D.3 Transition probabilities

$$\begin{Vmatrix} 32/33 & 0.93/33 & 0.07/33 \\ 0.06/8 & 7/8 & 0.94/8 \\ 0.4/24 & 0.6/24 & 23/24 \end{Vmatrix}$$

Appendix E: The Source Code

The source code of the software agent in C# is given in this section.

The agent has to be customized in order to be linked to a data source. The code given has 3 different input options. It can be linked to a text file, the tracking table of the hospital model built (a portion of the table is given in Table E.1 for reference. Consultation times are taken as the performance metric) or a tracking table that contains location data of a single tag collected at fixed intervals (a portion of the table is given in Table E.2 for reference). It can be run with or without data cleaning option if the third data source is chosen. The text file has to contain the process times of the event that is monitored, one value in each line.

The cleaning algorithm is set up for the RTLS accuracy distributions given in Table E.3 and transition probabilities given in Table E.4. Annotations are provided throughout the code. Two screenshots of the agent are given in Figure E.1 and Figure E.2 during analysis of the same data with and without the cleaning option respectively. The performance metric is taken as the stay length at location D, which comes from a normal distribution initially and then decreases following a learning curve. Notice that the agent could capture the behavior of the system when cleaning was performed on the data while analysis with no cleaning was completely inaccurate.

Table E.1 Tracking table of the hospital unit model

Record_ID	RFID_NO	Location_ID	Time_Stamp
210393	125	Consultation Room 2	5/20/2009 2:58
210394	3	Treatment Room 1	5/20/2009 2:58
210395	128	Consultation Room 2	5/20/2009 2:59
210396	117	Treatment Room 1	5/20/2009 3:00
210397	124	Treatment Room 1	5/20/2009 3:01
210398	5	Corridor	5/20/2009 3:04
210399	122	Corridor	5/20/2009 3:04
210400	127	Corridor	5/20/2009 3:04
210401	127	Doctors Room	5/20/2009 3:05
210402	5	Waiting Room	5/20/2009 3:05

Table E.2 Tracking table of the single tag model

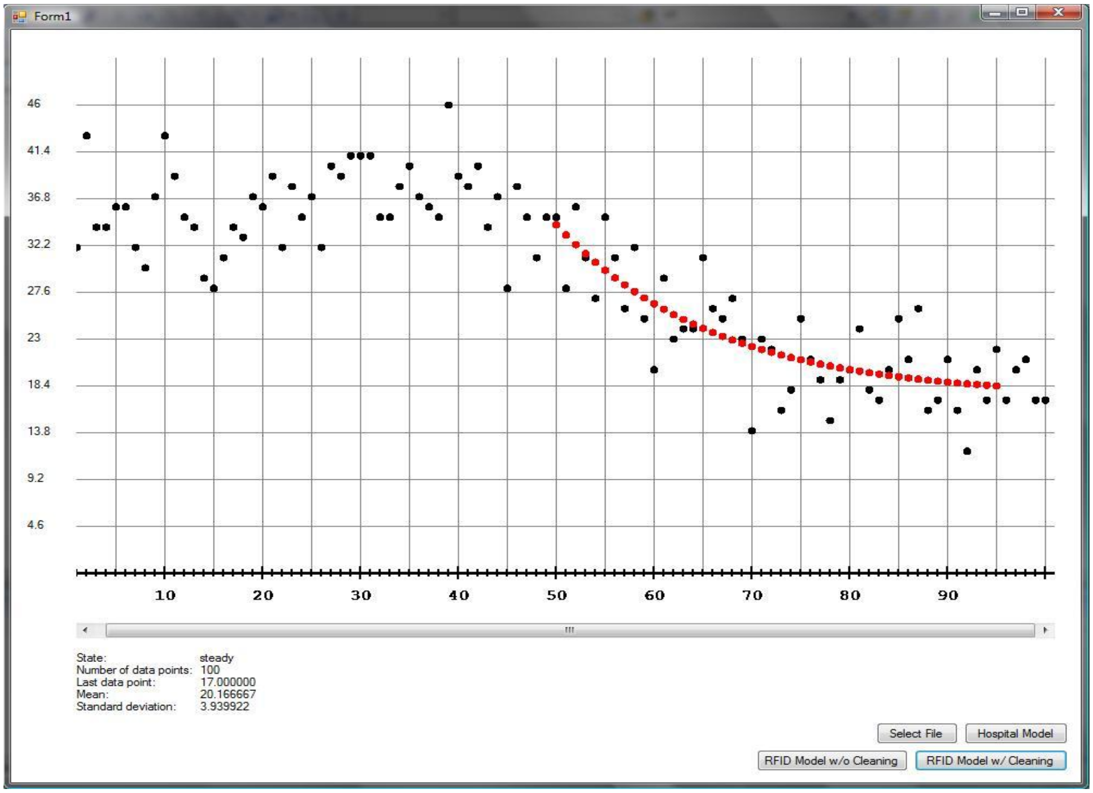
ID	Tag_ID	Location_ID	Time_stamp
377978	1	A	5/23/2009 12:00:10 AM
377979	1	B	5/23/2009 12:00:20 AM
377980	1	A	5/23/2009 12:00:30 AM
377981	1	B	5/23/2009 12:00:40 AM
377982	1	C	5/23/2009 12:00:50 AM
377983	1	A	5/23/2009 12:01:00 AM
377984	1	A	5/23/2009 12:01:10 AM
377985	1	A	5/23/2009 12:01:20 AM
377986	1	A	5/23/2009 12:01:30 AM
377987	1	B	5/23/2009 12:01:40 AM
377988	1	A	5/23/2009 12:01:50 AM
377989	1	A	5/23/2009 12:02:00 AM

Table E.3 Assumed RTLS accuracy

		Reported Location				
Actual Location		A	B	C	D	E
	A	0.8	0.08	0.08	0.02	0.02
	B	0.02	0.8	0.08	0.08	0.02
	C	0.02	0.02	0.8	0.08	0.08
	D	0.08	0.02	0.02	0.8	0.08
	E	0.08	0.08	0.02	0.02	0.8

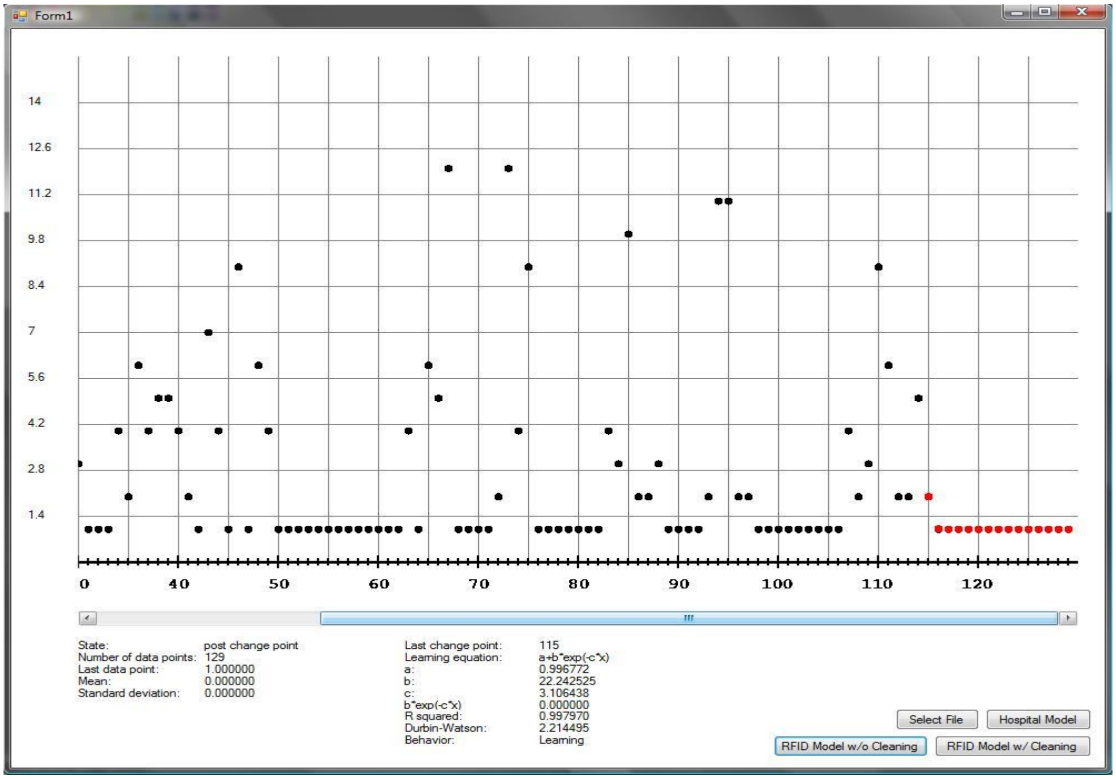
Table E.4 Assumed transition probabilities

$$\begin{array}{c||ccccc||}
 & A & B & C & D & E \\
A & 0.944444 & 0.022222 & 0.022222 & 0.011111 & 0 \\
B & 0 & 0.966667 & 0.026667 & 0.006667 & 0 \\
C & 0.008333 & 0 & 0.983333 & 0 & 0.008333 \\
D & 0 & 0.023333 & 0 & 0.966667 & 0.01 \\
E & 0.003333 & 0 & 0 & 0.013333 & 0.983333
\end{array}$$



State: steady
Number of data points: 100
Last data point: 17.000000
Mean: 20.166667
Standard deviation: 3.939922

Figure E.1 Software agent running with single tag data with cleaning



State: post change point
Number of data points: 129
Last data point: 1.000000
Mean: 0.000000
Standard deviation: 0.000000

Last change point: 115
Learning equation: a+b*exp(-c*x)
a: 0.996772
b: 22.242525
c: 3.106438
b*exp(-c*x) 0.000000
R squared: 0.997970
Durbin-Watson: 2.214495
Behavior: Learning

Figure E.2 Software agent running with single tag data without cleaning

The Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;
using System.Threading;
using System.Data.OleDb;

namespace Realtime_Performance_Monitor
{
    public partial class Form1 : Form
    {
        /*
         * The loading of a data source triggers analysis and plot.
This
         * happens at the function "Go". The best way to walk
         * through this code is to read that event first. The function
         * "watch" which includes the logic to perform the analysis
should
         * be read. The rest of the code contains supporting fuctions
most
         * of which include annotations. Annotations for plotting
         * functions are not given.
         */

        public double[] input = new double[100];
        public double[] y = new double[100];
        public string inputfile;
        public int numberofRows = 0;
        public int AnalyzedRows = 0;
        public int plottedRows = 0;
        public int i,j,k,l,m,n;
        public string state = "start up            ";
        public double mean, stdev;
        public int csWindowSize=30;
        public System.Timers.Timer updateTimer = new
System.Timers.Timer();
        public int lastChangePoint;
        public LM fitter1 = new LM();
        public LM fitter2 = new LM();
        public LC[] learningCurves = new LC[0];
        public bool model1Converged, model2Converged;
        public LC[] tempLC = new LC[0];
        public StreamWriter sw;
        public int method;

        public Graphics gShow;
        public Bitmap show;
        public Graphics gLC;
        public Bitmap graphLC;
        public Graphics g;
        public Bitmap graph = new Bitmap(900, 600);
        public Pen borderpen = new Pen(Color.Black, 3);
        public Pen gridpen = new Pen(Color.Gray, 1);
        public Pen graphLine = new Pen(Color.Black, 2);
        public SolidBrush pointFiller = new SolidBrush(Color.Black);
        public SolidBrush learningPointFiller = new
SolidBrush(Color.Red);
        public double largestData = 0;
        public double scale = 0;
        public bool needscroll=false;
        public Font labelFont = new Font("Courier New", 12,
FontStyle.Regular);

        delegate void updateGraphCallBack();

        public double[,] assumedTransitionMatrix = new double[5, 5];
        public double[,] assumedRTLSAccuracy = new double[5, 5];
        public string[] rawData = new string[100];
        public string[] cleanData = new string[100];
        public int numberOfReadings=0, numberOfCleanReadings = 0,
numberOfMinedReadings = 0;
        public int cleaningWindowSize;
        public int numberOfLocations = 5;
        public double[] probs = new double[5];
        public int[] output = new int[1];

        public Form1()
        {
            InitializeComponent();
            setupgraph();
            setupCleaning();
        }

        public long factorial(long number)
        {
            long answer = 1;
            while (number > 0)
            {
                answer *= number;
                number--;
            }
            return answer;
        }

        public int count(int index, string location)
        {
            /*
             * This function counts the number of readings from a given
             * location in the given analysis window. "index" specifies
             * the first reading in the considered window.
             */

            int result = 0;
            for (i = 0; i < 15; i++)
            {
                if (rawData[index + i] == location)
                {
                    result++;
                }
            }
            return result;
        }

        public double PWgivenL(int index, string location)
        {
            /*
             * This function calculates P(W|L) for a given location L
for
             * the scenario considered. P(W|L) comes from a multinomial
             * distribution.
             */

            if (location == "A")
            {
                return factorial(cleaningWindowSize) /
factorial(count(index, "A")) / factorial(count(index, "B")) /
factorial(count(index, "C")) / factorial(count(index, "D")) /
factorial(count(index, "E"))
                    * Math.Pow(assumedRTLSAccuracy[0, 0], count(index,
"A")) * Math.Pow(assumedRTLSAccuracy[0, 1], count(index, "B")) *
Math.Pow(assumedRTLSAccuracy[0, 2], count(index, "C")) *
Math.Pow(assumedRTLSAccuracy[0, 3], count(index, "D")) *
Math.Pow(assumedRTLSAccuracy[0, 4], count(index, "E"));
            }
            else if (location == "B")
            {
                return factorial(cleaningWindowSize) /
factorial(count(index, "A")) / factorial(count(index, "B")) /
factorial(count(index, "C")) / factorial(count(index, "D")) /
factorial(count(index, "E"))
                    * Math.Pow(assumedRTLSAccuracy[1, 0], count(index,
"A")) * Math.Pow(assumedRTLSAccuracy[1, 1], count(index, "B")) *
Math.Pow(assumedRTLSAccuracy[1, 2], count(index, "C")) *
Math.Pow(assumedRTLSAccuracy[1, 3], count(index, "D")) *
Math.Pow(assumedRTLSAccuracy[1, 4], count(index, "E"));
            }
            else if (location == "C")
            {
                return factorial(cleaningWindowSize) /
factorial(count(index, "A")) / factorial(count(index, "B")) /
factorial(count(index, "C")) / factorial(count(index, "D")) /
factorial(count(index, "E"))
                    * Math.Pow(assumedRTLSAccuracy[2, 0], count(index,
"A")) * Math.Pow(assumedRTLSAccuracy[2, 1], count(index, "B")) *
Math.Pow(assumedRTLSAccuracy[2, 2], count(index, "C")) *
Math.Pow(assumedRTLSAccuracy[2, 3], count(index, "D")) *
Math.Pow(assumedRTLSAccuracy[2, 4], count(index, "E"));
            }
            else if (location == "D")
            {
                return factorial(cleaningWindowSize) /
factorial(count(index, "A")) / factorial(count(index, "B")) /
factorial(count(index, "C")) / factorial(count(index, "D")) /
factorial(count(index, "E"))
                    * Math.Pow(assumedRTLSAccuracy[3, 0], count(index,
"A")) * Math.Pow(assumedRTLSAccuracy[3, 1], count(index, "B")) *
Math.Pow(assumedRTLSAccuracy[3, 2], count(index, "C")) *
Math.Pow(assumedRTLSAccuracy[3, 3], count(index, "D")) *
Math.Pow(assumedRTLSAccuracy[3, 4], count(index, "E"));
            }
            else if (location == "E")
            {
                return factorial(cleaningWindowSize) /
factorial(count(index, "A")) / factorial(count(index, "B")) /
factorial(count(index, "C")) / factorial(count(index, "D")) /
factorial(count(index, "E"))
                    * Math.Pow(assumedRTLSAccuracy[4, 0], count(index,
"A")) * Math.Pow(assumedRTLSAccuracy[4, 1], count(index, "B")) *
Math.Pow(assumedRTLSAccuracy[4, 2], count(index, "C")) *
Math.Pow(assumedRTLSAccuracy[4, 3], count(index, "D")) *
Math.Pow(assumedRTLSAccuracy[4, 4], count(index, "E"));
            }
            else return 0;
        }

        public int findMax(double[] inputArray)
        {
            /*
             * This function simply picks the largest element of an
array.
             * It is used to compare the conditional probabilities of
the
             * tag's being at locations. The deduced location of the
tag
             * is the location L that gives the highest P(L|W).
             */

            for (i = 0; i < inputArray.GetLength(0); i++)
            {
                for (j = 0; j < inputArray.GetLength(0); j++)
                {
                    if (inputArray[i] < inputArray[j])
                    {
                        break;
                    }
                }
                if (j == inputArray.GetLength(0))
                {
                    return i;
                }
            }
            return 10;
        }

        public void donotClean()
        {
            /*
             * This function is used to bypass the cleaning operation
             * when the user chooses to do so.
             */

            numberOfCleanReadings = numberOfReadings;
            cleanData = new string[numberOfCleanReadings];
            for (i = 0; i < numberOfCleanReadings; i++)
            {
                cleanData[i] = rawData[i];
            }
        }

        public void clean()
        {
            /*
             * This function cleans the raw data. It takes the raw data
             * from the array "rawData" and stores the cleaned data in
the
             * array "cleanData". It calculates P(L|W) for each
location L
             * which are stored in the array "probs". Afterwards, these
             * conditional probabilities are compared and the location
             * that has the largest P(L|W) is stored at the next
position
             * of "cleanData".
             */

            if (numberOfReadings >= cleaningWindowSize)
            {
                string[] temp = new string[numberOfReadings -
cleaningWindowSize + 1];
                for (i = 0; i < numberOfCleanReadings; i++)
                {
                    temp[i] = cleanData[i];
                }
                cleanData = temp;

                while (numberOfCleanReadings - 1 + cleaningWindowSize <
numberOfReadings)
                {
                    if (numberOfCleanReadings == 0)
                    {
                        probs[0] = PWgivenL(0, "A") *
assumedTransitionMatrix[0, 0] / (PWgivenL(0, "A") *
assumedTransitionMatrix[0, 0] + PWgivenL(0, "B") *
assumedTransitionMatrix[0, 1] + PWgivenL(0, "C") *
assumedTransitionMatrix[0, 2] + PWgivenL(0, "D") *
assumedTransitionMatrix[0, 3] + PWgivenL(0, "E") *
assumedTransitionMatrix[0, 4]);
                        probs[1] = PWgivenL(0, "B") *
assumedTransitionMatrix[0, 1] / (PWgivenL(0, "A") *
assumedTransitionMatrix[0, 0] + PWgivenL(0, "B") *
assumedTransitionMatrix[0, 1] + PWgivenL(0, "C") *
assumedTransitionMatrix[0, 2] + PWgivenL(0, "D") *
assumedTransitionMatrix[0, 3] + PWgivenL(0, "E") *
assumedTransitionMatrix[0, 4]);
                        probs[2] = PWgivenL(0, "C") *
assumedTransitionMatrix[0, 2] / (PWgivenL(0, "A") *
assumedTransitionMatrix[0, 0] + PWgivenL(0, "B") *
```

```
assumedTransitionMatrix[0, 1] + PWgivenL(0, "C") *
assumedTransitionMatrix[0, 2] + PWgivenL(0, "D") *
assumedTransitionMatrix[0, 3] + PWgivenL(0, "E") *
assumedTransitionMatrix[0, 4]);
                        probs[3] = PWgivenL(0, "D") *
assumedTransitionMatrix[0, 3] / (PWgivenL(0, "A") *
assumedTransitionMatrix[0, 0] + PWgivenL(0, "B") *
assumedTransitionMatrix[0, 1] + PWgivenL(0, "C") *
assumedTransitionMatrix[0, 2] + PWgivenL(0, "D") *
assumedTransitionMatrix[0, 3] + PWgivenL(0, "E") *
assumedTransitionMatrix[0, 4]);
                        probs[4] = PWgivenL(0, "E") *
assumedTransitionMatrix[0, 4] / (PWgivenL(0, "A") *
assumedTransitionMatrix[0, 0] + PWgivenL(0, "B") *
assumedTransitionMatrix[0, 1] + PWgivenL(0, "C") *
assumedTransitionMatrix[0, 2] + PWgivenL(0, "D") *
assumedTransitionMatrix[0, 3] + PWgivenL(0, "E") *
assumedTransitionMatrix[0, 4]);

                        if (findMax(probs) == 0)
                        {
                            cleanData[numberOfCleanReadings] = "A";
                        }
                        else if (findMax(probs) == 1)
                        {
                            cleanData[numberOfCleanReadings] = "B";
                        }
                        else if (findMax(probs) == 2)
                        {
                            cleanData[numberOfCleanReadings] = "C";
                        }
                        else if (findMax(probs) == 3)
                        {
                            cleanData[numberOfCleanReadings] = "D";
                        }
                        else if (findMax(probs) == 4)
                        {
                            cleanData[numberOfCleanReadings] = "E";
                        }

                        numberOfCleanReadings++;
                    }
                    else
                    {
                        if (cleanData[numberOfCleanReadings - 1] ==
"A")
                        {
                            probs[0] = PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[0, 0] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[0, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[0, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[0, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[0, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[0, 4]);
                            probs[1] = PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[0, 1] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[0, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[0, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[0, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[0, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[0, 4]);
                            probs[2] = PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[0, 2] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[0, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[0, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[0, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[0, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[0, 4]);
                            probs[3] = PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[0, 3] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[0, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[0, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[0, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[0, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[0, 4]);
                            probs[4] = PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[0, 4] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[0, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[0, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[0, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[0, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[0, 4]);
                        }
                        else if (cleanData[numberOfCleanReadings - 1]
== "B")
                        {
                            probs[0] = PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[1, 0] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[1, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[1, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[1, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[1, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[1, 4]);
                            probs[1] = PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[1, 1] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[1, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[1, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[1, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[1, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[1, 4]);
                            probs[2] = PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[1, 2] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[1, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[1, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[1, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[1, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[1, 4]);
                            probs[3] = PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[1, 3] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[1, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[1, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[1, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[1, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[1, 4]);
```

```
                            probs[4] = PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[1, 4] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[1, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[1, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[1, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[1, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[1, 4]);
                        }
                        else if (cleanData[numberOfCleanReadings - 1]
== "C")
                        {
                            probs[0] = PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[2, 0] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[2, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[2, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[2, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[2, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[2, 4]);
                            probs[1] = PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[2, 1] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[2, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[2, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[2, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[2, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[2, 4]);
                            probs[2] = PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[2, 2] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[2, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[2, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[2, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[2, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[2, 4]);
                            probs[3] = PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[2, 3] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[2, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[2, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[2, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[2, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[2, 4]);
                            probs[4] = PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[2, 4] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[2, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[2, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[2, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[2, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[2, 4]);
                        }
                        else if (cleanData[numberOfCleanReadings - 1]
== "D")
                        {
                            probs[0] = PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[3, 0] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[3, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[3, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[3, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[3, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[3, 4]);
```

```
                                        probs[1] = PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[3, 1] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[3, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[3, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[3, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[3, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[3, 4]);
                                        probs[2] = PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[3, 2] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[3, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[3, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[3, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[3, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[3, 4]);
                                        probs[3] = PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[3, 3] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[3, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[3, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[3, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[3, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[3, 4]);
                                        probs[4] = PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[3, 4] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[3, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[3, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[3, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[3, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[3, 4]);
                                    }
                                    else if (cleanData[numberOfCleanReadings - 1]
== "E")
                                    {
                                        probs[0] = PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[4, 0] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[4, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[4, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[4, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[4, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[4, 4]);
                                        probs[1] = PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[4, 1] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[4, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[4, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[4, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[4, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[4, 4]);
                                        probs[2] = PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[4, 2] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[4, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[4, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[4, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[4, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[4, 4]);
                                        probs[3] = PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[4, 3] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[4, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[4, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[4, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[4, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[4, 4]);
                            probs[4] = PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[4, 4] / (PWgivenL(numberOfCleanReadings,
"A") * assumedTransitionMatrix[4, 0] + PWgivenL(numberOfCleanReadings,
"B") * assumedTransitionMatrix[4, 1] + PWgivenL(numberOfCleanReadings,
"C") * assumedTransitionMatrix[4, 2] + PWgivenL(numberOfCleanReadings,
"D") * assumedTransitionMatrix[4, 3] + PWgivenL(numberOfCleanReadings,
"E") * assumedTransitionMatrix[4, 4]);
                        }
                        if (findMax(probs) == 0)
                        {
                            cleanData[numberOfCleanReadings] = "A";
                        }
                        else if (findMax(probs) == 1)
                        {
                            cleanData[numberOfCleanReadings] = "B";
                        }
                        else if (findMax(probs) == 2)
                        {
                            cleanData[numberOfCleanReadings] = "C";
                        }
                        else if (findMax(probs) == 3)
                        {
                            cleanData[numberOfCleanReadings] = "D";
                        }
                        else if (findMax(probs) == 4)
                        {
                            cleanData[numberOfCleanReadings] = "E";
                        }

                        numberOfCleanReadings++;
                    }
                }
            }
        }

        public void mine(string location)
        {
            /*
             * This function derives the stay lengths at a given
             * location L from the array "cleanData" where the tracking
             * data is stored. The stay lengths are stored in 'output'
             */

            while (numberOfMinedReadings < numberOfCleanReadings)
            {
                if (numberOfMinedReadings == 0)
                {
                    if (cleanData[0] == location)
                    {
                        output[0]++;
                    }
                }
                else
                {
                    if (cleanData[numberOfMinedReadings - 1] ==
location && cleanData[numberOfMinedReadings] == location)
                    {
                        output[output.GetLength(0) - 1]++;
                    }
                    else if (cleanData[numberOfMinedReadings - 1] ==
location && cleanData[numberOfMinedReadings] != location)
                    {

                    }
                    else if (cleanData[numberOfMinedReadings - 1] !=
location && cleanData[numberOfMinedReadings] == location)
                    {
                        if (output[output.GetLength(0) - 1] == 0)
                        {

                        }
                        else
                        {
                            int[] temp = new int[output.GetLength(0) +
1];

                            for (i = 0; i < output.GetLength(0); i++)
                            {
                                temp[i] = output[i];
                            }
                            output = temp;
                        }
                        output[output.GetLength(0) - 1] = 1;
                    }
                }
                numberOfMinedReadings++;
            }
        }

        public void setupCleaning()
        {
            /*
             * This function sets up the arrays that hold the
transition
             * probabilities and accuracy distributions.
             */

            cleaningWindowSize = 15;

            for (i = 0; i < 5; i++)
            {
                for (j = 0; j < 5; j++)
                {
                    if (i == j)
                    {
                        assumedRTLSAccuracy[i, j] = 0.8;
                    }
                    else if (j == ((i + 1) % 5))
                    {
                        assumedRTLSAccuracy[i, j] = 0.08;
                    }
                    else if (j == ((i + 2) % 5))
                    {
                        assumedRTLSAccuracy[i, j] = 0.08;
                    }
                    else
                    {
                        assumedRTLSAccuracy[i, j] = 0.02;
                    }
                }
            }
            assumedTransitionMatrix[0, 0] = 17.0 / 18;
            assumedTransitionMatrix[0, 1] = 0.4 / 18;
            assumedTransitionMatrix[0, 2] = 0.4 / 18;
            assumedTransitionMatrix[0, 3] = 0.2 / 18;
            assumedTransitionMatrix[0, 4] = 0;

            assumedTransitionMatrix[1, 0] = 0;
            assumedTransitionMatrix[1, 1] = 29.0 / 30;
            assumedTransitionMatrix[1, 2] = 0.8 / 30;
            assumedTransitionMatrix[1, 3] = 0.2 / 30;
            assumedTransitionMatrix[1, 4] = 0 / 30;

            assumedTransitionMatrix[2, 0] = 0.5 / 60;
            assumedTransitionMatrix[2, 1] = 0;
            assumedTransitionMatrix[2, 2] = 59.0 / 60;
            assumedTransitionMatrix[2, 3] = 0;
            assumedTransitionMatrix[2, 4] = 0.5 / 60;

            assumedTransitionMatrix[3, 0] = 0;
            assumedTransitionMatrix[3, 1] = 0.7 / 30;
            assumedTransitionMatrix[3, 2] = 0;
            assumedTransitionMatrix[3, 3] = 29.0 / 30;
            assumedTransitionMatrix[3, 4] = 0.3 / 30;

            assumedTransitionMatrix[4, 0] = 0.2 / 60;
            assumedTransitionMatrix[4, 1] = 0;
            assumedTransitionMatrix[4, 2] = 0;
            assumedTransitionMatrix[4, 3] = 0.8 / 60;
            assumedTransitionMatrix[4, 4] = 59.0 / 60;
        }

        public void readRawData()
        {
            /*
             * This function reads the tracking data from the database
             * table. This function is used when the data from the 
single
             * tag model is to be analyzed.
             */

            numberOfReadings = 0;

            string conntectionstring = 
"Provider=Microsoft.ACE.OLEDB.12.0;Data 
Source=C:\\Users\\Cenk\\Desktop\\Healthcare\\simulation model\\data 
cleaning example\\Example.accdb;Persist Security Info=False;";
            OleDbConnection conn = new 
OleDbConnection(conntectionstring);
```

```
            OleDbCommand selectcommand = new OleDbCommand("select 
location_ID from tracking order by id", conn);
            OleDbDataAdapter da = new OleDbDataAdapter(selectcommand);
            DataSet ds = new DataSet();
            numberOfReadings = da.Fill(ds, "tracking");
            rawData = new string[numberOfReadings];
            for (j = 0; j < numberOfReadings; j++)
            {
                rawData[j] = 
ds.Tables["tracking"].Rows[j][0].ToString();
            }
        }

        double findcp(double[] y,double mean,double stdev,int first)
        {
            /*
             * Fahmy's changepoint detection method
             * Inputs
             * y: data array
             * mean: steady state distribution mean
             * stdev: steady state distribution standard deviation
             * first: first data point in the window
             * Output: chi square statistic for Fahmy's method.
             * Calculation is done using a window of size 20. All of 
the
             * data is passed to the function. Input "first" determines
             * the window. For example, if there are 100 data points 
and
             * the variable first is 60, then the analysis is done for
             * the window 60-79.
             */

            int i;
            double chisquare=0;
            double stt = 665;
            double tavg = 10.5;
            double div = ((20.0 - 10.5) * (20.0 - 10.5)) / stt;
            double beta = 0;
            double betaDenom = 0;
            double alpha = 0;
            double windowmean = 0;

            for (i = 0; i < 20; i++)
            {
                windowmean += y[i+first] / 20;
            }

            for (i = 0; i < 20; i++)
            {
                beta += (i + 1 - tavg) * (y[i+first] - windowmean);
                betaDenom += (i + 1 - tavg) * (i + 1 - tavg);
            }
            beta = beta / betaDenom;
            alpha = windowmean - beta * tavg;

            chisquare = Math.Pow((mean - (alpha + beta * 20.0)) / 
(Math.Sqrt(stdev * stdev * (0.05 + div))), 2);
```

```
            return chisquare;
        }

        bool chisquare(double[] y, double mean, double stdev, int
first)
        {
            /*
             * Chi square distribution fit test
             * Inputs:
             * y: data array
             * mean: sample mean
             * stdev: sample standard deviation
             * first: first data point in the window
             * Output: true if H0 (sample comes from normal
distribution)
             * is not rejected.
             * All of the data is passed to the function. Input "first"
             * determines the window. There are two options for sample
             * size: 30 and 50. Variable "csWindowSize" holds this
number.
             */

            if (csWindowSize == 30)
            {

                int i;
                int[] observed = new int[5];
                double statistic;

                for (i = 0; i < 5; i++)
                {
                    observed[i] = 0;
                }

                for (i = first; i < 30 + first; i++)
                {
                    if (y[i] < -0.841621234 * stdev + mean)
                    {
                        observed[0]++;
                    }
                    else if (y[i] >= -0.841621234 * stdev + mean &&
y[i] < -0.253347103 * stdev + mean)
                    {
                        observed[1]++;
                    }
                    else if (y[i] >= -0.253347103 * stdev + mean &&
y[i] < 0.253347103 * stdev + mean)
                    {
                        observed[2]++;
                    }
                    else if (y[i] >= 0.253347103 * stdev + mean && y[i]
< 0.841621234 * stdev + mean)
                    {
                        observed[3]++;
                    }
                    else if (y[i] >= 0.841621234 * stdev + mean)
```

```
                        {
                            observed[4]++;
                        }
                    }
                    statistic = (Math.Pow(observed[0] - 6, 2) +
Math.Pow(observed[1] - 6, 2) + Math.Pow(observed[2] - 6, 2) +
Math.Pow(observed[3] - 6, 2) + Math.Pow(observed[4] - 6, 2)) / 6;

                    /*
                    chisquare with two degree of freedom at alpha = 0.05
                    */

                    if (statistic < 5.99)
                    {
                        return true;
                    }
                    else
                    {
                        return false;
                    }
                }
                else if (csWindowSize == 50)
                {
                    int i;
                    int[] observed = new int[8];
                    double statistic;

                    for (i = 0; i < 8; i++)
                    {
                        observed[i] = 0;
                    }

                    for (i = first; i < 50 + first; i++)
                    {
                        if (y[i] < -1.15034938 * stdev + mean)
                        {
                            observed[0]++;
                        }
                        else if (y[i] >= -1.15034938 * stdev + mean && y[i]
< -0.67448975 * stdev + mean)
                        {
                            observed[1]++;
                        }
                        else if (y[i] >= -0.67448975 * stdev + mean && y[i]
< -0.318639364 * stdev + mean)
                        {
                            observed[2]++;
                        }
                        else if (y[i] >= -0.318639364 * stdev + mean &&
y[i] < 0 * stdev + mean)
                        {
                            observed[3]++;
                        }
                        else if (y[i] >= 0 * stdev + mean && y[i] <
0.318639364 * stdev + mean)
                        {
                            observed[4]++;
                        }
                        else if (y[i] >= 0.318639364 * stdev + mean && y[i]
< 0.67448975 * stdev + mean)
                        {
                            observed[5]++;
                        }
                        else if (y[i] >= 0.67448975 * stdev + mean && y[i]
< 1.15034938 * stdev + mean)
                        {
                            observed[6]++;
                        }
                        else if (y[i] >= 1.15034938 * stdev + mean)
                        {
                            observed[7]++;
                        }
                    }
                    statistic = (Math.Pow(observed[0] - 6.25, 2) +
Math.Pow(observed[1] - 6.25, 2) + Math.Pow(observed[2] - 6.25, 2) +
Math.Pow(observed[3] - 6.25, 2) + Math.Pow(observed[4] - 6.25, 2) +
Math.Pow(observed[5] - 6.25, 2) + Math.Pow(observed[6] - 6.25, 2) +
Math.Pow(observed[7] - 6.25, 2)) / 6.25;
                    /*
                    chisquare with five degree of freedom at alpha = 0.05
                    */
                    if (statistic < 11.1)
                    {
                        return true;
                    }
                    else
                    {
                        return false;
                    }
                }
                return false;
            }

            bool portmanteau(double[] x, double mean, int first)
            {
                int i,j;
                double num = 0, div = 0, statistic = 0;

                for (i = first; i < csWindowSize + first; i++)
                {
                    div += (x[i] - mean) * (x[i] - mean);
                }

                if (csWindowSize == 50)
                {
                    for (i = 1; i < 11; i++)
                    {
                        num = 0;
                        for (j = first; j < first + csWindowSize - i; j++)
                        {
                            num += (x[j] - mean) * (x[j + i] - mean);
                        }
                        statistic += csWindowSize * (csWindowSize + 2) *
(num / div) * (num / div) / (csWindowSize - i);
```

```
                }
                /*
                 * alpha = 0.1 dof=10 16
                 * alpha = 0.5 dof=10 9.342
                 */

                if (statistic <= 16)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            else if (csWindowSize == 30)
            {
                for (i = 1; i < 6; i++)
                {
                    num = 0;
                    for (j = first; j < first + csWindowSize - i; j++)
                    {
                        num += (x[j] - mean) * (x[j + i] - mean);
                    }
                    statistic += csWindowSize * (csWindowSize + 2) *
(num / div) * (num / div) / (csWindowSize - i);
                }
                /*
                 * alpha = 0.1 dof=5 9.236
                 * alpha = 0.5 dof=5 4.351
                 */

                if (statistic <= 4.351)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            else
            {
                return false;
            }
        }

        double findmean(double[] x, int first)
        {
            double mean = 0;
            int i;
            for (i = first; i < csWindowSize + first; i++)
            {
                mean += x[i] / csWindowSize;
            }
            return mean;
        }
        double findstdev(double[] x, double mean, int first)
        {
            double stdev = 0;
            int i;
            for (i = first; i < csWindowSize + first; i++)
            {
                stdev += Math.Pow(x[i] - mean, 2) / (csWindowSize-1);
            }
            stdev = Math.Pow(stdev, 0.5);
            return stdev;
        }

        void readfromfile()
        {
            /*
             * This function is used when data from a text file is to 
be
             * analyzed.
             * The data from the file is loaded into the array "input".
             * This is a temporary storage. Analysis is actually 
performed
             * on the array "y". "input" is a dynamic array which
             * increases its capacity when necessary. "numberofRows" is
             * the number of data points read at a single call of the
             * function.
             */

            System.IO.StreamReader sr = new 
System.IO.StreamReader(inputfile);
            string line;
            numberofRows = 0;
            while ((line = sr.ReadLine()) != null)
            {
                if (input.GetLength(0) == numberofRows)
                {
                    double[] tempinput = new double[input.GetLength(0) 
* 2];
                    for (j = 0; j < input.GetLength(0); j++)
                    {
                        tempinput[j] = input[j];
                    }
                    input = tempinput;
                }
                input[numberofRows] = Double.Parse(line);
                numberofRows++;
            }
            sr.Close();
        }

        void readfromlocaldatabase()
        {
            /*
             * This function is used when data from the hospital unit
             * model is to be analyzed.
             */
            numberofRows = 0;

            string conntectionstring =
"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Users\\Cenk\\Desktop\\Healthcare\\simulation
model\\Hospital.accdb;Persist Security Info=False;";
            OleDbConnection conn = new
OleDbConnection(conntectionstring);
            OleDbCommand selectcommand = new OleDbCommand("select
record_id, datediff('n', previous_time_stamp,time_stamp) " +
                "from " +
                "(select
t1.record_id,t1.rfid_no,t1.location_id,t1.time_stamp, " +
                "(select top 1 record_id from tracking t2 where
t1.record_id>t2.record_id and t1.rfid_no=t2.rfid_no order by
t2.record_id desc) as previous_record_id, " +
                "(select top 1 location_id from tracking t2 where
t1.record_id>t2.record_id and t1.rfid_no=t2.rfid_no order by
t2.record_id desc) as previous_location_id, " +
                "(select top 1 time_stamp from tracking t2 where
t1.record_id>t2.record_id and t1.rfid_no=t2.rfid_no order by
t2.record_id desc) as previous_time_stamp " +
                "from tracking t1 " +
                "where rfid_no>=100 and rfid_no<=120 " +
                "order by t1.record_id) nt " +
                "where previous_location_id = 'equipment cleaning'",
conn);
            OleDbDataAdapter da = new OleDbDataAdapter(selectcommand);
            DataSet ds = new DataSet();
            numberofRows = da.Fill(ds, "tracking");
            input = new double[numberofRows];
            for (j = 0; j < numberofRows; j++)
            {

input[j]=Double.Parse(ds.Tables["tracking"].Rows[j][1].ToString());
            }
        }

        void watch()
        {
            /*
             * This is the main function that does all the analysis.
Data
             * source is checked every 5000ms. If the number of data
points
             * read is different from the number processed, analysis
             * begins.
             * The variable 'method' sets the source of the data
according
             * to the button clicked in the interface.
             */

            while (true)
            {
                if (method == 1)
                {
                    readfromfile();
```

```
    while (numberofRows == AnalyzedRows)
    {
        Thread.Sleep(5000);
        readfromfile();
    }
}
else if (method == 2)
{
    readfromlocaldatabase();
    while (numberofRows == AnalyzedRows)
    {
        Thread.Sleep(5000);
        readfromlocaldatabase();
    }
}
else if (method == 3)
{
    readRawData();
    donotClean();
    mine("D");
    numberofRows = output.GetLength(0) - 1;
    input = new double[numberofRows];
    for (j = 0; j < numberofRows; j++)
    {
        input[j] = output[j];
    }
    while (numberofRows == AnalyzedRows)
    {
        Thread.Sleep(5000);
        readRawData();
        donotClean();
        mine("D");
        numberofRows = output.GetLength(0) - 1;
        input = new double[numberofRows];
        for (j = 0; j < numberofRows; j++)
        {
            input[j] = output[j];
        }
    }
}

else if (method == 4)
{
    readRawData();
    clean();
    mine("D");
    numberofRows = output.GetLength(0) - 1;
    input = new double[numberofRows];
    for (j = 0; j < numberofRows; j++)
    {
        input[j] = output[j];
    }
    while (numberofRows == AnalyzedRows)
    {
        Thread.Sleep(5000);
        readRawData();
        clean();
```

```
                        mine("D");
                        numberofRows = output.GetLength(0) - 1;
                        input = new double[numberofRows];
                        for (j = 0; j < numberofRows; j++)
                        {
                            input[j] = output[j];
                        }
                    }
                }

                while (AnalyzedRows < numberofRows)
                {
                    /*
                     * "y" is also a dynamic array that increases its
                     * capacity if necessary. The data is fed into this
                     * array one row at a time from "input" so that
even
                     * if more than 1 data points are read, the
analysis
                     * is done for 1 data point at a time. For each new
                     * data point, only one type of analysis is
performed,
                     * depending on the state variable. There are 3
                     * states: "start up", "steady state" and "post
                     * change point".
                     */

                    if (y.GetLength(0) == AnalyzedRows)
                    {
                        double[] tempy = new double[y.GetLength(0) *
2];
                        for (j = 0; j < y.GetLength(0); j++)
                        {
                            tempy[j] = y[j];
                        }
                        y = tempy;
                    }
                    y[AnalyzedRows] = input[AnalyzedRows];

                    /*
                     * if the program is in start up  phase, no
parameters
                     * have been set. If enough points to perform a chi
                     * square distribution test accumulate, the test is
                     * done. If we fail to reject H0 that "the data
comes
                     * from a normal distribution" the state is updated
to
                     * "steady state"
                     */

                    if (state == "start up            ")
                    {
                        if (AnalyzedRows + 1 >= csWindowSize)
                        {
                            mean = findmean(y, AnalyzedRows -
csWindowSize + 1);
```

```
                            stdev = findstdev(y, mean, AnalyzedRows -
csWindowSize + 1);
                            if (chisquare(y, mean, stdev, AnalyzedRows
- csWindowSize + 1) && portmanteau(y, mean, AnalyzedRows - csWindowSize
+ 1))
                            {
                                state = "steady              ";
                            }
                        }
                    }

                    /*
                     * If steady state parameters have been set, change
                     * points are seeked. If a window gives a chi
square
                     * statistic that is higher than the upper control
                     * limit, which is 7.65 for window size 20, then a
                     * record for a new learning curve is created. The
                     * change point is recorded as "lastChangePoint".
                     */

                    else if (state == "steady              ")
                    {
                        i = AnalyzedRows - 20 + 1;

                        if (findcp(y, mean, stdev, i) > 7.65)
                        {
                            while (findcp(y, mean, stdev, i) -
findcp(y, mean, stdev, i - 1) > 0)
                            {
                                i--;
                            }

                            tempLC = new LC[learningCurves.GetLength(0)
+ 1];
                            for (j = 0; j <
learningCurves.GetLength(0); j++)
                            {
                                tempLC[j] = new LC();
                                tempLC[j] = learningCurves[j];
                            }
                            tempLC[tempLC.GetLength(0) - 1] = new LC();
                            learningCurves = tempLC;

                            mean = 0;
                            stdev = 0;
                            lastChangePoint = i + 20 - 1;
//lastChangePoint += 3;
                            learningCurves[learningCurves.GetLength(0)
- 1].lastchangepoint = lastChangePoint;
                            state = "post change point   ";
                            AnalyzedRows = lastChangePoint;
                            break;
                        }
                    }

                    /*
```

```
                     * If a change points has been observed, two new
                     * calculations are performed: fitting of a
learning
                     * curve and searching for new steady state
variables.
                     * Searching for new steady state variables is done
                     * only if enough points are accumulated. Fitting
of a
                     * learning curve is performed only if at least 5
                     * data points are accumulated.
                     */

                    else if (state == "post change point   ")
                    {
                        if (AnalyzedRows - lastChangePoint >=
csWindowSize - 1)
                        {
                            i = AnalyzedRows - csWindowSize + 1;

                            mean = findmean(y, i);
                            stdev = findstdev(y, mean, i);
                            if (chisquare(y, mean, stdev,
i)&&portmanteau(y,mean,i))
                            {
                                state = "steady              ";
                            }
                        }

                        if (AnalyzedRows - lastChangePoint >= 4)
                        {
                            /*
                             * The code segment below employes the 'two
                             * part fitting algorithm to fit a step
shift
                             * followed by learning to the data.
                             */

                            fitlearningcurve();

                            learningCurves[learningCurves.GetLength(0)
- 1].behavior = "Learning                ";

                            if
(learningCurves[learningCurves.GetLength(0) - 1].last != AnalyzedRows)
                            {
                                twopartfit();

learningCurves[learningCurves.GetLength(0) - 1].behavior = "Learning
after step shift";
                            }

                            else if
(learningCurves[learningCurves.GetLength(0) - 1].r2 < 0.2 ||
learningCurves[learningCurves.GetLength(0)-1].durbinwatson>2.5 ||
learningCurves[learningCurves.GetLength(0)-1].durbinwatson<1.5
||Math.Abs(learningCurves[learningCurves.GetLength(0) -
1].transientTerm) < 0.001 * learningCurves[learningCurves.GetLength(0)
- 1].a)
                            {
                                twopartfit();

learningCurves[learningCurves.GetLength(0) - 1].behavior = "Learning
after step shift";

                                if
(learningCurves[learningCurves.GetLength(0) - 1].r2 < 0.2 ||
learningCurves[learningCurves.GetLength(0) - 1].durbinwatson > 2.5 ||
learningCurves[learningCurves.GetLength(0) - 1].durbinwatson < 1.5 ||
Math.Abs(learningCurves[learningCurves.GetLength(0) - 1].transientTerm)
< 0.001 * learningCurves[learningCurves.GetLength(0) - 1].a)
                                {
                                    fitlearningcurve();

learningCurves[learningCurves.GetLength(0) - 1].behavior = "Step shift
";
                                }
                            }
                        }
                    }
                    AnalyzedRows++;
                    printoutput();
                }
            }
        }

        void fitlearningcurve()
        {
            /*
             * This function employes fitter1 and fitter2 to fit two
             * different learning curves after the change point.
             * fitter1 and fitter2 are instances of the class 'LM'
which
             * contains the function for nonlinear fitting.
             */

            fitter1.nDataPoints = AnalyzedRows - lastChangePoint + 1;
            fitter1.x = new double[fitter1.nDataPoints];
            fitter1.y = new double[fitter1.nDataPoints];

            fitter2.nDataPoints = AnalyzedRows - lastChangePoint + 1;
            fitter2.x = new double[fitter2.nDataPoints];
            fitter2.y = new double[fitter2.nDataPoints];

            for (i = lastChangePoint; i <= AnalyzedRows; i++)
            {
                fitter1.x[i - lastChangePoint] = i + 1 -
lastChangePoint;
                fitter1.y[i - lastChangePoint] = y[i];

                fitter2.x[i - lastChangePoint] = i + 1 -
lastChangePoint;
                fitter2.y[i - lastChangePoint] = y[i];
            }
            fitter1.setinitialparam();
            fitter2.setinitialparam();

            model1Converged = fitter1.regress();
            model2Converged = fitter2.regress();

            /*
             * fitter1 and fitter2 are curve fitting
             * objects. fitter 1 is for the model 'y=a+b*exp(-cx)'
             * and fitter2 is for 'y=a+bx^-c'.
             * In the above lines, data is fed into
             * members of these objects and the necessary
             * functions are called. In the below lines,
             * if both models converge, the one with the
             * smaller error is recorded as a learning
             * curve. If only one converge, that one is
             * recorded. Variables like the type of the curve,
             * parameters, the first and the last data
             * point on the curve are stored.
             */


            if (model1Converged && model2Converged)
            {
                if (fitter1.a > 0 && fitter2.a > 0)
                {
                    if (fitter1.e <= fitter2.e)
                    {

setcurrentcp(learningCurves[learningCurves.GetLength(0) - 1], fitter1);
                    }
                    else if (fitter1.e > fitter2.e)
                    {

setcurrentcp(learningCurves[learningCurves.GetLength(0) - 1], fitter2);
                    }
                }
                else if (fitter1.a > 0)
                {

setcurrentcp(learningCurves[learningCurves.GetLength(0) - 1], fitter1);
                }
                else if (fitter2.a > 0)
                {

setcurrentcp(learningCurves[learningCurves.GetLength(0) - 1], fitter2);
                }
            }
            else if (model1Converged && fitter1.a>0)
            {
                setcurrentcp(learningCurves[learningCurves.GetLength(0)
- 1], fitter1);
            }
            else if (model2Converged && fitter2.a>0)
            {
```

```
        setcurrentcp(learningCurves[learningCurves.GetLength(0)
- 1], fitter2);
      }
      else
      {
        //none converged
      }
    }
    void twopartfit()
    {
      /*
       * This functions utilizes fitter1 and fitter2 to fit
       * two segment lines to the data. All possible joining
points
       * are considered for the two segments. The configuration
that
       * minimizes the total sum of squares is selected as the
       * best fit.
       */

      fitlearningcurve();

      double pastfirstpartsse = 0;
      double firstpartsse = 0, firstpartmean = 0;
      LC currentLC = new LC();

      for (j = lastChangePoint; AnalyzedRows - j + 1 > 4; j++)
      {
        fitter1.nDataPoints = AnalyzedRows - j + 1;
        fitter1.x = new double[fitter1.nDataPoints];
        fitter1.y = new double[fitter1.nDataPoints];

        fitter2.nDataPoints = AnalyzedRows - j + 1;
        fitter2.x = new double[fitter2.nDataPoints];
        fitter2.y = new double[fitter2.nDataPoints];

        for (i = j; i <= AnalyzedRows; i++)
        {
          fitter1.x[i - j] = i + 1 - j;
          fitter1.y[i - j] = y[i];

          fitter2.x[i - j] = i + 1 - j;
          fitter2.y[i - j] = y[i];
        }

        fitter1.setinitialparam();
        fitter2.setinitialparam();

        model1Converged = fitter1.regress();
        model2Converged = fitter2.regress();

        firstpartsse = 0;
        firstpartmean = 0;

        for (i = lastChangePoint; i < j; i++)
        {
          firstpartmean += y[i] / (j - lastChangePoint);
```

```
                }
                for (i = lastChangePoint; i < j; i++)
                {
                    firstpartsse += (y[i] - firstpartmean) * (y[i] -
firstpartmean);
                }

                if (model1Converged && model2Converged)
                {
                    if (fitter1.a > 0 && fitter2.a > 0)
                    {
                        if (fitter1.e <= fitter2.e)
                        {
                            setcurrentcp(currentLC, fitter1);
                            currentLC.first = j;
                        }
                        else if (fitter1.e > fitter2.e)
                        {
                            setcurrentcp(currentLC, fitter2);
                            currentLC.first = j;
                        }
                    }
                    else if (fitter1.a > 0)
                    {
                        setcurrentcp(currentLC, fitter1);
                        currentLC.first = j;
                    }
                    else if(fitter2.a>0)
                    {
                        setcurrentcp(currentLC, fitter2);
                        currentLC.first = j;
                    }
                }
                else if (model1Converged && fitter1.a > 0)
                {
                    setcurrentcp(currentLC, fitter1);
                    currentLC.first = j;
                }
                else if (model2Converged && fitter2.a > 0)
                {
                    setcurrentcp(currentLC, fitter2);
                    currentLC.first = j;
                }
                else
                {
                    //none converged
                }

                if ((learningCurves[learningCurves.GetLength(0) -
1].last!=AnalyzedRows || currentLC.sse + firstpartsse <
learningCurves[learningCurves.GetLength(0) - 1].sse + pastfirstpartsse)
&& currentLC.first == j)
                {
                    learningCurves[learningCurves.GetLength(0) - 1].a =
currentLC.a;
                    learningCurves[learningCurves.GetLength(0) - 1].b =
currentLC.b;
```

```
                    learningCurves[learningCurves.GetLength(0) - 1].c =
currentLC.c;
                    learningCurves[learningCurves.GetLength(0) -
1].model = currentLC.model;
                    learningCurves[learningCurves.GetLength(0) -
1].first = currentLC.first;
                    learningCurves[learningCurves.GetLength(0) -
1].last = currentLC.last;
                    learningCurves[learningCurves.GetLength(0) - 1].r2
= currentLC.r2;
                    learningCurves[learningCurves.GetLength(0) -
1].durbinwatson = currentLC.durbinwatson;
                    learningCurves[learningCurves.GetLength(0) - 1].sse
= currentLC.sse;
                    learningCurves[learningCurves.GetLength(0) -
1].nDataPoints = currentLC.nDataPoints;
                    pastfirstpartsse = firstpartsse;
                }
            }
        }

        void printoutput()
        {
            /*
             * This function prints an output file that includes the
             * results of calculations at every iteration of the main
             * loop.
             */

            sw = File.AppendText(Directory.GetCurrentDirectory() +
"\\output.txt");

            if (AnalyzedRows - lastChangePoint >= 5 &&
learningCurves.GetLength(0) > 0&&state=="post change point   ")
            {
                if (learningCurves[learningCurves.GetLength(0) -
1].model == 1)
                {
                    sw.WriteLine(state.ToString() + "\t" +
AnalyzedRows.ToString() + "\t" + y[AnalyzedRows -
1].ToString("0.000000") +
                    "\t" + mean.ToString("0.000000") + "\t" +
stdev.ToString("0.000000") + "\t" +
                    (learningCurves[learningCurves.GetLength(0) -
1].lastchangepoint + 1).ToString() + "\t" +
                    (learningCurves[learningCurves.GetLength(0) -
1].first + 1).ToString() + "\t" + "\t" +
                    "a+b*exp(-c*x)" + "\t" +
                    learningCurves[learningCurves.GetLength(0) -
1].a.ToString("0.000000") + "\t" +
learningCurves[learningCurves.GetLength(0) - 1].b.ToString("0.000000")
+
                    "\t" + learningCurves[learningCurves.GetLength(0) -
1].c.ToString("0.000000") + "\t" +
                    (learningCurves[learningCurves.GetLength(0) - 1].b
* Math.Exp(-learningCurves[learningCurves.GetLength(0) - 1].c *
(learningCurves[learningCurves.GetLength(0) - 1].last -
learningCurves[learningCurves.GetLength(0) - 1].first + 
1))).ToString("0.000000") + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].r2.ToString("0.000000") + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].durbinwatson.ToString("0.000000") + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].behavior);

                    sw.Close();
                }
                else if (learningCurves[learningCurves.GetLength(0) - 
1].model == 2)
                {

                    sw.WriteLine(state.ToString() + "\t" + 
AnalyzedRows.ToString() + "\t" + y[AnalyzedRows - 
1].ToString("0.000000") + 
                    "\t" + mean.ToString("0.000000") + "\t" + 
stdev.ToString("0.000000") + "\t" + 
                    (learningCurves[learningCurves.GetLength(0) - 
1].lastchangepoint + 1).ToString() + "\t" + 
                    (learningCurves[learningCurves.GetLength(0) - 
1].first + 1).ToString() + "\t" + "\t" + 
                    "a+b*x^c      " + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].a.ToString("0.000000") + "\t" + 
learningCurves[learningCurves.GetLength(0) - 1].b.ToString("0.000000") 
+ 
                    "\t" + learningCurves[learningCurves.GetLength(0) - 
1].c.ToString("0.000000") + "\t" + 
                    (learningCurves[learningCurves.GetLength(0) - 1].b 
* Math.Pow((learningCurves[learningCurves.GetLength(0) - 1].last - 
learningCurves[learningCurves.GetLength(0) - 1].first + 1), 
learningCurves[learningCurves.GetLength(0) - 
1].c)).ToString("0.000000") + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].r2.ToString("0.000000") + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].durbinwatson.ToString("0.000000") + "\t" + 
                    learningCurves[learningCurves.GetLength(0) - 
1].behavior);

                    sw.Close();
                }
                else
                {
                    sw.Close();
                }
            }
            else
            {
                sw.WriteLine(state.ToString() + "\t" + 
AnalyzedRows.ToString() + "\t" + y[AnalyzedRows - 
1].ToString("0.000000") + 
                    "\t" + mean.ToString("0.000000") + "\t" + 
stdev.ToString("0.000000") + "\t");
```

```
                sw.Close();
            }
        }

        /*
         * Below are the functions that update the graphical user
         * interface.
         */

        void printlabels()
        {
            if ((largestData * 0.1).ToString().Length > 5)
            {
                label1.Text = (largestData *
0.1).ToString().Substring(0, 5);
            }
            else
            {
                label1.Text = (largestData * 0.1).ToString();
            }
            if ((largestData * 0.2).ToString().Length > 5)
            {
                label2.Text = (largestData *
0.2).ToString().Substring(0, 5);
            }
            else
            {
                label2.Text = (largestData * 0.2).ToString();
            }
            if ((largestData * 0.3).ToString().Length > 5)
            {
                label3.Text = (largestData *
0.3).ToString().Substring(0, 5);
            }
            else
            {
                label3.Text = (largestData * 0.3).ToString();
            }
            if ((largestData * 0.4).ToString().Length > 5)
            {
                label4.Text = (largestData *
0.4).ToString().Substring(0, 5);
            }
            else
            {
                label4.Text = (largestData * 0.4).ToString();
            }
            if ((largestData * 0.5).ToString().Length > 5)
            {
                label5.Text = (largestData *
0.5).ToString().Substring(0, 5);
            }
            else
            {
                label5.Text = (largestData * 0.5).ToString();
            }
            if ((largestData * 0.6).ToString().Length > 5)
            {
                label6.Text = (largestData *
0.6).ToString().Substring(0, 5);
            }
            else
            {
                label6.Text = (largestData * 0.6).ToString();
            }
            if ((largestData * 0.7).ToString().Length > 5)
            {
                label7.Text = (largestData *
0.7).ToString().Substring(0, 5);
            }
            else
            {
                label7.Text = (largestData * 0.7).ToString();
            }
            if ((largestData * 0.8).ToString().Length > 5)
            {
                label8.Text = (largestData *
0.8).ToString().Substring(0, 5);
            }
            else
            {
                label8.Text = (largestData * 0.8).ToString();
            }
            if ((largestData * 0.9).ToString().Length > 5)
            {
                label9.Text = (largestData *
0.9).ToString().Substring(0, 5);
            }
            else
            {
                label9.Text = (largestData * 0.9).ToString();
            }
            if ((largestData * 1.0).ToString().Length > 5)
            {
                label10.Text = (largestData *
1.0).ToString().Substring(0, 5);
            }
            else
            {
                label10.Text = (largestData * 1.0).ToString();
            }
        }
        void setupgraph()
        {
            graph = new Bitmap(900, 600);
            g = Graphics.FromImage(graph);
            this.BackColor = Color.White;

            for (l = 0; l < 100; l = l + 5)
            {
                g.DrawLine(gridpen, l * 9, 0, l * 9, 550);
            }
            for (l = 0; l < 10; l++)
            {
                g.DrawLine(gridpen, 0, 500 - l * 50, 900, 500 - l * 
50);
            }
            for (l = 1; l < 100; l++)
            {
                g.DrawLine(graphLine, l * 9, 546, l * 9, 554);
            }
            for (l = 10; l < 100; l = l + 10)
            {
                g.DrawLine(graphLine, l * 9, 543, l * 9, 557);
            }
            for (l = 10; l < 100; l = l + 10)
            {
                g.DrawString(l.ToString(), labelFont, pointFiller, l*9-
12, 565);
            }

            g.DrawLine(borderpen, 0, 550, 900, 550);
            g.DrawLine(borderpen, 0, 0, 0, 550);

            pictureBox1.Size = graph.Size;
            pictureBox1.Image = graph;
        }
        void updategraph()
        {
            if (AnalyzedRows > plottedRows)
            {
                while(plottedRows<AnalyzedRows)
                {
                    if (y[plottedRows] > largestData)
                    {
                        largestData = y[plottedRows];
                        scale = 500.0 / largestData;
                        setupgraph();
                        plottedRows = 0;
                        printlabels();
                    }
                    if (plottedRows < 99)
                    {
                        float xCor = (float)((plottedRows + 1) * 9.0 - 
4.0);
                        float yCor = (float)(550.0 - y[plottedRows] * 
scale - 4.0);

                        g.FillEllipse(pointFiller, xCor, yCor, 8, 8);
                        plottedRows++;
                    }
                    else
                    {
                        if (panel1.AutoScrollPosition.X == (plottedRows 
- 99) * -9)
                        {
                            needscroll = true;
                        }
                        Bitmap temp =new 
Bitmap(graph.Width+9,graph.Height);
                        Graphics gTemp = Graphics.FromImage(temp);
                        gTemp.DrawImage(graph, 0, 0);

                        for (l = 0; l < 10; l++)
                        {
                            gTemp.DrawLine(gridpen, graph.Width, 500 -
l * 50, graph.Width+9, 500 - l * 50);
                        }
                        if (graph.Width % 45 == 0)
                        {
                            gTemp.DrawLine(gridpen, graph.Width, 0,
graph.Width, 550);
                            //gTemp.DrawLine(graphLine, graph.Width,
543, graph.Width, 557);
                        }

                        if ((graph.Width-36) % 90 == 0)
                        {
                            gTemp.DrawString((plottedRows-
3).ToString(), labelFont, pointFiller, graph.Width-54, 565);
                        }
                        if (graph.Width % 90 == 0)
                        {
                            gTemp.DrawLine(gridpen, graph.Width, 0,
graph.Width, 550);
                            gTemp.DrawLine(graphLine, graph.Width, 543,
graph.Width, 557);
                        }

                        gTemp.DrawLine(borderpen, graph.Width, 550,
graph.Width + 9, 550);
                        gTemp.DrawLine(graphLine, graph.Width, 546,
graph.Width, 554);

                        float xCor = (float)((plottedRows + 1) * 9.0 -
4.0);
                        float yCor = (float)(550.0 - y[plottedRows] *
scale - 4.0);
                        gTemp.FillEllipse(pointFiller, xCor, yCor, 8,
8);

                        graph = temp;
                        plottedRows++;
                    }
                }

                graphLC = new Bitmap(graph.Width, graph.Height);
                gLC = Graphics.FromImage(graphLC);

                for (l = 0; l < learningCurves.GetLength(0); l++)
                {
                    for (m = 0; m <= learningCurves[l].last -
learningCurves[l].first; m++)
                    {
                        float xCor = (float)((learningCurves[l].first +
1 + m) * 9.0 - 4.0);
                        float yCor = 0;
                        if (learningCurves[l].model == 1)
                        {
                            yCor = (float)(550 - (learningCurves[l].a +
learningCurves[l].b * Math.Exp(-learningCurves[l].c * (m + 1))) * scale
- 4.0);
                        }
                        else if (learningCurves[l].model == 2)
                        {
                            yCor = (float)(550 - (learningCurves[l].a +
learningCurves[l].b * Math.Pow(m + 1, learningCurves[l].c)) * scale -
4.0);
                        }
                        gLC.FillEllipse(learningPointFiller, xCor,
yCor, 8, 8);
                    }
                }

                show = new Bitmap(graph.Width, graph.Height);
                gShow = Graphics.FromImage(show);
                gShow.DrawImage(graph, 0, 0);
                gShow.DrawImage(graphLC, 0, 0);

                pictureBox1.Size = show.Size;
                pictureBox1.Image = show;

                if (needscroll)
                {
                    panel1.AutoScrollPosition = new Point((plottedRows
- 99) * 9, 0);
                    needscroll = false;
                }

                if (state == "start up            ")
                {
                    label21.Visible = false;
                    label22.Visible = false;
                    label23.Visible = false;
                    label24.Visible = false;
                    label25.Visible = false;
                    label26.Visible = false;
                    label27.Visible = false;
                    label28.Visible = false;
                    label29.Visible = false;
                    label30.Visible = false;
                    label31.Visible = false;
                    label32.Visible = false;
                    label33.Visible = false;
                    label34.Visible = false;
                    label35.Visible = false;
                    label36.Visible = false;
                    label37.Visible = false;
                    label38.Visible = false;

                    label11.Text = AnalyzedRows.ToString();
                    label12.Text = state.ToString();
                    label13.Text = mean.ToString("0.000000");
                    label14.Text = stdev.ToString("0.000000");
                    label15.Text = y[AnalyzedRows -
1].ToString("0.000000");
                }
                else if (state == "steady              ")
                {
                    label21.Visible = false;
                    label22.Visible = false;
                    label23.Visible = false;
                    label24.Visible = false;
                    label25.Visible = false;
                    label26.Visible = false;
                    label27.Visible = false;
                    label28.Visible = false;
                    label29.Visible = false;
                    label30.Visible = false;
                    label31.Visible = false;
                    label32.Visible = false;
                    label33.Visible = false;
                    label34.Visible = false;
                    label35.Visible = false;
                    label36.Visible = false;
                    label37.Visible = false;
                    label38.Visible = false;

                    label11.Text = AnalyzedRows.ToString();
                    label12.Text = state.ToString();
                    label13.Text = mean.ToString("0.000000");
                    label14.Text = stdev.ToString("0.000000");
                    label15.Text = y[AnalyzedRows -
1].ToString("0.000000");
                }
                else if (state == "post change point   ")
                {
                    label11.Text = AnalyzedRows.ToString();
                    label12.Text = state.ToString();
                    label13.Text = mean.ToString("0.000000");
                    label14.Text = stdev.ToString("0.000000");
                    label15.Text = y[AnalyzedRows -
1].ToString("0.000000");

                    label21.Visible = true;
                    label22.Visible = true;
                    label23.Visible = true;
                    label24.Visible = true;
                    label25.Visible = true;
                    label26.Visible = true;
                    label27.Visible = true;
                    label28.Visible = true;
                    label29.Visible = true;
                    label30.Visible = true;
                    label31.Visible = true;
                    label32.Visible = true;
                    label33.Visible = true;
                    label34.Visible = true;
                    label35.Visible = true;
                    label36.Visible = true;
                    label37.Visible = true;
                    label38.Visible = true;

                    if (AnalyzedRows - lastChangePoint >= 5 &&
learningCurves.GetLength(0) > 0)
                    {
                        if (learningCurves[learningCurves.GetLength(0)
- 1].model == 1)
                        {
                            label26.Text="b*exp(-c*x)";
                            label29.Text = "a+b*exp(-c*x)";
                            label33.Text =
(learningCurves[learningCurves.GetLength(0) - 1].b * Math.Exp(-
learningCurves[learningCurves.GetLength(0) - 1].c *
(learningCurves[learningCurves.GetLength(0) - 1].last -
learningCurves[learningCurves.GetLength(0) - 1].first +
1))).ToString("0.000000");
                        }
                        else if
(learningCurves[learningCurves.GetLength(0) - 1].model == 2)
                        {
                            label26.Text = "b*x^c";
                            label29.Text = "a+b*x^c";
                            label33.Text =
(learningCurves[learningCurves.GetLength(0) - 1].b *
Math.Pow((learningCurves[learningCurves.GetLength(0) - 1].last -
learningCurves[learningCurves.GetLength(0) - 1].first + 1),
learningCurves[learningCurves.GetLength(0) -
1].c)).ToString("0.000000");
                        }
                        label30.Text =
learningCurves[learningCurves.GetLength(0) - 1].a.ToString("0.000000");
                        label31.Text =
learningCurves[learningCurves.GetLength(0) - 1].b.ToString("0.000000");
                        label32.Text =
learningCurves[learningCurves.GetLength(0) - 1].c.ToString("0.000000");
                        label28.Text =
(learningCurves[learningCurves.GetLength(0) - 1].lastchangepoint +
1).ToString();
                        label34.Text =
learningCurves[learningCurves.GetLength(0) -
1].r2.ToString("0.000000");
                        label36.Text =
learningCurves[learningCurves.GetLength(0) -
1].durbinwatson.ToString("0.000000");
                        label38.Text =
learningCurves[learningCurves.GetLength(0) - 1].behavior;
                    }
                }
            }
        }

        /*
         * This is where everything starts. When a data source is
linked,
         * some parameters for regression objects are set. Then two
         * threads are called. "tWatch" does the analysis and "tPlot"
         * plots the data. They share the data and some variables so
that
         * every analyzed data point can be plotted. Plotting is done
         * every 1 second and data is read every 5 seconds,
         * triggering an analysis loop if new data could be read. See
the
         * function "watch".
         */

        public void Go()
        {
            File.Delete(Directory.GetCurrentDirectory() +
"\\output.txt");
            sw = File.AppendText(Directory.GetCurrentDirectory() +
"\\output.txt");
            sw.WriteLine("State               " + "\t" + "Rows" + "\t"
+ "Last Data" + "\t" + "Mean      " + "\t" + "Stdev      " + "\t" +
                "Last CP" + "\t" + "LC Start" + "\t" + "Learning M" +
"\t" + "a       " + "\t" + "b         " + "\t" + "c         " + "\t" +
                "Transient T" + "\t" + "R2        " + "\t" + "Durbin-
Watson");
            sw.Close();

            fitter1.nIterations = 10000;
            fitter1.nParameters = 3;
            fitter1.sensitivity = 0.000001;
            fitter1.ssest = 0.005;
            fitter1.model = 1;

            fitter2.nIterations = 10000;
            fitter2.nParameters = 3;
            fitter2.sensitivity = 0.000001;
            fitter2.ssest = 0.005;
            fitter2.model = 2;

            Thread tWatch = new Thread(watch);
            tWatch.IsBackground = true;
            tWatch.Start();

            Thread tPlot = new Thread(plot);
            tPlot.IsBackground = true;
            tPlot.Start();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            openFileDialog1.InitialDirectory =
Directory.GetCurrentDirectory();
            openFileDialog1.Filter = "Text Files|*.txt";
            openFileDialog1.Title = "Select a Data File";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                inputfile = openFileDialog1.FileName;
            }
            method = 1;
            Go();
        }

        void plot()
        {
            while (true)
            {
                updateGraphCallBack u = new
updateGraphCallBack(updategraph);
                this.Invoke(u);
                //updategraph();
                Thread.Sleep(1000);
            }
        }

        void setcurrentcp(LC destination, LM source)
        {
            destination.a = source.a;
            destination.b = source.b;
            destination.c = source.c;
            destination.model = source.model;
            destination.first = lastChangePoint;
            destination.last = AnalyzedRows;
            destination.r2 = source.R2;
            destination.durbinwatson = source.durbinwatson;
            destination.sse = source.sse;
            destination.nDataPoints = source.nDataPoints;
            if (destination.model == 1)
            {
                destination.transientTerm = destination.b * Math.Exp(-
destination.c * (destination.last - destination.first + 1));
            }
            else if (destination.model == 2)
            {
                destination.transientTerm = destination.b *
Math.Pow((destination.last - destination.first + 1), destination.c);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            method = 2;
            Go();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            method = 4;
            Go();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            method = 3;
            Go();
        }
    }
    public class LM
    {
        /*
         * This is the class for nonlinear regression.
         */

        public int nDataPoints;
        public int nParameters;
        public int nIterations;
        public int nUpdateJ;
        public double lamda, det, sensitivity;
        public double a, b, c, e, a1, b1, c1, e1;
        public int i, j, k, n;
        public int model;
        public double R2,durbinwatson;

        public double[,] J, H, H2, H2inv;
        public double[] x, y, d, d1, J2;

        public double[] logx, logy;
        public double logxmean, logymean, beta, betadenom, alpha,
ssest;

        public double sse, sst;

        public LM()
        {
            model = -1;
            sensitivity = 0.000001;
            ssest = 0.005;
        }

        /*
         * Below are the auxillary functions used in nonlinear
regression
         * calculations.
         */

        public void calJ()
        {
            if (model == 1)
            {
                for (i = 0; i < nDataPoints; i++)
                {
                    J[i, 0] = -1;
                    J[i, 1] = -Math.Exp(-c * x[i]);
                    J[i, 2] = b * x[i] * Math.Exp(-c * x[i]);
                }
            }
            else if (model == 2)
            {
                for (i = 0; i < nDataPoints; i++)
                {
                    J[i, 0] = -1;
                    J[i, 1] = -Math.Pow(x[i], c);
                    J[i, 2] = -b * Math.Pow(x[i], c) * Math.Log(x[i]);
                }
            }
        }

        public void cald()
        {
            if (model == 1)
            {
                for (i = 0; i < nDataPoints; i++)
                {
                    d[i] = y[i] - (a + b * Math.Exp(-c * x[i]));
                }
            }
            else if (model == 2)
            {
                for (i = 0; i < nDataPoints; i++)
                {
                    d[i] = y[i] - (a + b * Math.Pow(x[i], c));
                }
            }
        }

        public void calH2inv()
        {
            if (model == 1 || model == 2)
            {
                det = (H2[0, 0] * (H2[1, 1] * H2[2, 2] - H2[1, 2] *
H2[2, 1]) -
                        H2[0, 1] * (H2[1, 0] * H2[2, 2] - H2[1, 2] *
H2[2, 0]) +
                        H2[0, 2] * (H2[1, 0] * H2[2, 1] - H2[1, 1] *
H2[2, 0])
                        );

                H2inv[0, 0] = (H2[1, 1] * H2[2, 2] - H2[1, 2] * H2[2,
1]) / det;
                H2inv[0, 1] = (H2[0, 2] * H2[2, 1] - H2[0, 1] * H2[2,
2]) / det;
                H2inv[0, 2] = (H2[0, 1] * H2[1, 2] - H2[0, 2] * H2[1,
1]) / det;
                H2inv[1, 0] = (H2[1, 2] * H2[2, 0] - H2[1, 0] * H2[2,
2]) / det;
                H2inv[1, 1] = (H2[0, 0] * H2[2, 2] - H2[0, 2] * H2[2,
0]) / det;
                H2inv[1, 2] = (H2[0, 2] * H2[1, 0] - H2[0, 0] * H2[1,
2]) / det;
                H2inv[2, 0] = (H2[1, 0] * H2[2, 1] - H2[1, 1] * H2[2,
0]) / det;
                H2inv[2, 1] = (H2[0, 1] * H2[2, 0] - H2[0, 0] * H2[2,
1]) / det;
                H2inv[2, 2] = (H2[0, 0] * H2[1, 1] - H2[0, 1] * H2[1,
0]) / det;
            }
        }

        public void calParam()
        {
            if (model == 1 || model == 2)
```

```
    {
        a1 = a;
        b1 = b;
        c1 = c;

        for (i = 0; i < nParameters; i++)
        {
            a1 -= H2inv[0, i] * J2[i];
            b1 -= H2inv[1, i] * J2[i];
            c1 -= H2inv[2, i] * J2[i];
        }
    }
}

public void cald1()
{
    if (model == 1)
    {
        for (i = 0; i < nDataPoints; i++)
        {
            d1[i] = y[i] - (a1 + b1 * Math.Exp(-c1 * x[i]));
        }
    }
    if (model == 2)
    {
        for (i = 0; i < nDataPoints; i++)
        {
            d1[i] = y[i] - (a1 + b1 * Math.Pow(x[i], c1));
        }
    }
}

public void setinitialparam()
{
    /*
     * This function sets the initial parameters for nonlinear
     * regression for the given data using the method developed
     * in the thesis.
     */

    logx = new double[nDataPoints];
    logy = new double[nDataPoints];

    beta = 0;
    betadenom = 0;
    alpha = 0;
    logxmean = 0;
    logymean = 0;

    for (i = 0; i < nDataPoints; i++)
    {
        logx[i] = Math.Log(x[i]);
        logy[i] = Math.Log(y[i]);
        logxmean += logx[i] / nDataPoints;
        logymean += logy[i] / nDataPoints;
    }
```

```
            for (i = 0; i < nDataPoints; i++)
            {
                beta += (logx[i] - logxmean) * (logy[i] - logymean);
                betadenom += (logx[i] - logxmean) * (logx[i] -
logxmean);
            }
            beta = beta / betadenom;
            alpha = logymean - beta * logxmean;

            if (model == 1)
            {
                a = Math.Exp(alpha) *
Math.Pow(Math.Ceiling(Math.Abs(beta / ssest)), beta);
                //b = Math.Pow(Math.Pow((Math.Exp(alpha) - a),
Math.Ceiling(Math.Abs(beta / ssest)) - 1) / (Math.Exp(alpha) *
Math.Pow(Math.Ceiling(Math.Abs(beta / ssest)) - 1, beta) - a), 1 /
(Math.Ceiling(Math.Abs(beta / ssest)) - 1 - 1));
                b = (Math.Exp(alpha) - a) * (Math.Exp(alpha) - a) /
(Math.Exp(alpha) * Math.Pow(2, beta) - a);
                c = -Math.Log((Math.Exp(alpha) - a) / b);
            }

            else if (model == 2)
            {
                a = Math.Exp(alpha) *
Math.Pow(Math.Ceiling(Math.Abs(beta / ssest)), beta);
                b = Math.Exp(alpha) - a;
                //c = Math.Log((Math.Exp(alpha) *
Math.Pow(Math.Ceiling(Math.Abs(beta / ssest)) - 1, beta) - a) / b,
Math.Ceiling(Math.Abs(beta / ssest)) - 1);
                c = Math.Log((Math.Exp(alpha) * Math.Pow(2, beta) - a)
/ b, 2);
            }
        }

        void calR2()
        {
            double average=0;
            sse = 0;
            sst = 0;

            for (i = 0; i < nDataPoints; i++)
            {
                average += y[i] / nDataPoints;
            }

            if (model == 1)
            {
                for (i = 0; i < nDataPoints; i++)
                {
                    sse += (a + b * Math.Exp(-c * (i+1)) - y[i]) * (a +
b * Math.Exp(-c * (i+1)) - y[i]);
                    sst += (y[i] - average) * (y[i] - average);
                }
            }
            else if (model == 2)
            {
                for (i = 0; i < nDataPoints; i++)
                {
                    sse += (a + b * Math.Pow(i+1, c) - y[i]) * (a + b *
Math.Pow(i+1, c) - y[i]);
                    sst += (y[i] - average) * (y[i] - average);
                }
            }
            R2 = 1 - sse / sst;
        }

        void caldurbinwatson()
        {
            double num = 0, div = 0;

            if (model == 1)
            {
                div+=Math.Pow( (a + b * Math.Exp(-c * (1)) - y[0]) ,2);
                for (i = 1; i < nDataPoints; i++)
                {
                    num += Math.Pow((a + b * Math.Exp(-c * (i + 1)) -
y[i]) - (a + b * Math.Exp(-c * (i)) - y[i - 1]), 2);
                    div += Math.Pow((a + b * Math.Exp(-c * (i + 1)) -
y[i]), 2);
                }
            }
            else if (model == 2)
            {
                div += Math.Pow((a + b * Math.Pow(1, c) - y[0]), 2);
                for (i = 1; i < nDataPoints; i++)
                {
                    num += Math.Pow((a + b * Math.Pow(i + 1, c) - y[i])
- (a + b * Math.Pow(i, c) - y[i - 1]), 2);
                    div += Math.Pow((a + b * Math.Pow(i + 1, c) -
y[i]), 2);
                }
            }
            durbinwatson = num / div;
        }

        public bool regress()
        {
            /*
             * This is the function that performs the iterations and
             * checks if convergence is achieved.
             */

            if (model == 1 || model == 2)
            {
                nParameters = 3;
            }

            lamda = 0.01;
            nUpdateJ = 1;
            J = new double[nDataPoints, nParameters];
            H = new double[nParameters, nParameters];
            H2 = new double[nParameters, nParameters];
            H2inv = new double[nParameters, nParameters];
```

```
d = new double[nDataPoints];
d1 = new double[nDataPoints];
J2 = new double[nDataPoints];

for (n = 0; n < nIterations; n++)
{
    if (nUpdateJ == 1)
    {
        calJ();
        cald();

        for (i = 0; i < nParameters; i++)
        {
            for (j = 0; j < nParameters; j++)
            {
                H[i, j] = 0;
                for (k = 0; k < nDataPoints; k++)
                {
                    H[i, j] += J[k, i] * J[k, j];
                }
            }
        }

        if (n == 0)
        {
            e = 0;
            for (i = 0; i < nDataPoints; i++)
            {
                e += d[i] * d[i];
            }
        }
    }

    for (i = 0; i < nParameters; i++)
    {
        for (j = 0; j < nParameters; j++)
        {
            H2[i, j] = 0;
        }
    }
    for (i = 0; i < nParameters; i++)
    {
        H2[i, i] = H[i, i] + lamda;
    }

    /*
    for (i = 0; i < nParameters; i++)
    {
        for (j = 0; j < nParameters; j++)
        {
            H2[i, j] = H[i, j];
        }
    }

    for (i = 0; i < nParameters; i++)
    {
```

```
                    H2[i, i] += lamda;
                }
                */
                calH2inv();

                for (i = 0; i < nParameters; i++)
                {
                    J2[i] = 0;
                }

                for (i = 0; i < nParameters; i++)
                {
                    for (j = 0; j < nDataPoints; j++)
                    {
                        J2[i] += J[j, i] * d[j];
                    }
                }

                calParam();

                cald1();

                e1 = 0;

                for (i = 0; i < nDataPoints; i++)
                {
                    e1 += d1[i] * d1[i] /*/ ((5 + 5 * Math.Exp(-(i + 5)
* 0.05)) )*/;
                }
                if (e1 < e)
                {
                    if ((e - e1) / e < sensitivity)
                    {
                        lamda = lamda / 10;
                        a = a1;
                        b = b1;
                        c = c1;
                        e = e1;
                        nUpdateJ = 1;
                        calR2();
                        caldurbinwatson();

                        return true;
                    }
                    lamda = lamda / 10;
                    a = a1;
                    b = b1;
                    c = c1;
                    e = e1;
                    nUpdateJ = 1;
                }
                else
                {
                    nUpdateJ = 0;
                    lamda = lamda * 10;
                }
            }
```

```
                return false;
            }
        }

        public class LC
        {
            /*
             * This is a data structure to record learning curves.
             */

            public LC()
            {

            }
            public int model;
            public double a,b,c,r2,durbinwatson,sse,transientTerm;
            public int first, last, nDataPoints, lastchangepoint;
            public string behavior;
        }
    }
```

Appendix F: The Simulation Model

The SIMAN code for the hospital unit model is given in this section.

The model consists of two sub-models. The main sub-model represents the hospital unit. In this sub-model, four types of entities, namely patients, doctors, nurses and equipment follow the workflow given in chapter 5.3.1 of the thesis. The locations of the tags are stored in an array named “Track” and are updated when tags move from one location to another. The second sub-model represents the RTLS where a single token loops through the logic every 10 seconds. This token writes the locations of the tags stored in the array “Track” into a database file along with the tag IDs and timestamps.

The RTLS logic has two important options. The first option enables maintaining a more compact database by writing only the readings that imply a change in the location of a tag. It can be turned on by setting the variable “CompactData” equal to 1. This option exploits the fact that unless a tag reports from a different location it can be assumed to be in the same location. This option should only be used under the assumption of perfect data because it imposes an aggregation on the tracking data which makes data cleaning using the proposed algorithm impossible. The second option adds errors to the tracking data according to distributions given by the user. It can be turned on by setting the variable “Errors” equal to 1.

The list of variables used in the model is given in Table F.1. “Track” stores the states (busy/idle) of the tags in the first column and locations in the second column for each tag ID. “Places” stores the true locations and other two locations that may be reported in case of a tracking error for each location ID. This table is set up by the user according to desired RTLS accuracy characteristics. A sample set up is given in Table F.2. Error probabilities are set in the module named “Check error probability”. “PreviousLocation” stores the one previous location that was reported for each tag in order to implement the data compacting option. Both “PreviousLocation” and “Track” have 128 rows for 128 total tags: 100 for patients, 20 for equipment, 6 for nurses and 2 for doctors.

Table F.1 List of variables

Variable	**Rows**	**Columns**
RFIDCounter	1	1
TreatmentRoom1NeedNurse	1	1
TreatmentRoom2NeedNurse	1	1
TreatmentRoom3NeedNurse	1	1
TreatmentRoom4NeedNurse	1	1
TreatmentRoom5NeedNurse	1	1
TreatmentRoom6NeedNurse	1	1
TreatmentRoom1NeedEquipment	1	1
TreatmentRoom2NeedEquipment	1	1
TreatmentRoom3NeedEquipment	1	1
TreatmentRoom4NeedEquipment	1	1
TreatmentRoom5NeedEquipment	1	1
TreatmentRoom6NeedEquipment	1	1
TreatmentRoom1NurseRequestTime	1	1
TreatmentRoom2NurseRequestTime	1	1
TreatmentRoom3NurseRequestTime	1	1
TreatmentRoom4NurseRequestTime	1	1
TreatmentRoom5NurseRequestTime	1	1
TreatmentRoom6NurseRequestTime	1	1
TreatmentRoom1EquipmentRequestTime	1	1
TreatmentRoom2EquipmentRequestTime	1	1
TreatmentRoom3EquipmentRequestTime	1	1
TreatmentRoom4EquipmentRequestTime	1	1
TreatmentRoom5EquipmentRequestTime	1	1
TreatmentRoom6EquipmentRequestTime	1	1
EquipmentCleaned	1	1
DoctorCounter	1	1
NurseCounter	1	1
EquipmentCounter	1	1
PatientCounter	1	1
ConsultationRoom1NurseRequestTime	1	1
ConsultationRoom1DoctorRequestTime	1	1
ConsultationRoom1NeedDoc	1	1
ConsultationRoom1NeedNurse	1	1

ConsultationRoom1Status	1	1
ConsultationRoom2DoctorRequestTime	1	1
ConsultationRoom2NurseRequestTime	1	1
ConsultationRoom2NeedDoc	1	1
ConsultationRoom2NeedNurse	1	1
ConsultationRoom2Status	1	1
TreatmentRoom1Status	1	1
TreatmentRoom2Status	1	1
TreatmentRoom3Status	1	1
TreatmentRoom4Status	1	1
TreatmentRoom5Status	1	1
TreatmentRoom6Status	1	1
cleaningmeantime	1	1
Track	128	2
TrackerCounter	1	1
Places	15	3
Errors	1	1
CompactData	1	1
PreviousLocation	128	1

Table F.2 A sample set up for "Places"

Location ID	True Location	Reporting Error 1	Reporting Error 2
1	Corridor	Corridor	Corridor
2	Register	Register	Register
3	Waiting Room	Consultation Room 1	Corridor
4	Consultation Room 1	Consultation Room 2	Corridor
5	Consultation Room 2	Consultation Room 1	Corridor
6	Nurses Room	Doctors Room	Corridor
7	Doctors Room	Nurses Room	Corridor
8	Treatment Room 1	Treatment Room 2	Corridor
9	Treatment Room 2	Treatment Room 3	Corridor
10	Treatment Room 3	Treatment Room 4	Corridor
11	Treatment Room 4	Treatment Room 5	Corridor
12	Treatment Room 5	Treatment Room 6	Corridor
13	Treatment Room 6	Treatment Room 1	Corridor
14	Equipment Room	Equipment Cleaning	Corridor
15	Equipment Cleaning	Equipment Room	Corridor

The Code

```
;
;
;     Model statements for module:  BasicProcess.Create 5 (Create Doctor)
;

302$          CREATE,
2,HoursToBaseTime(0.0),Doctor:HoursToBaseTime(1),1:NEXT(303$);

303$          ASSIGN:        Create Doctor.NumberOut=Create Doctor.NumberOut +
1:NEXT(0$);


;
;
;     Model statements for module:  BasicProcess.Assign 58 (Assign doctor ID)
;
0$            ASSIGN:        DoctorCounter=DoctorCounter+1:
                             RFID=126+DoctorCounter:NEXT(239$);


;
;
;     Model statements for module:  BasicProcess.Assign 101 (Make tag status in
use for doctors)
;
239$          ASSIGN:        Track(RFID,1)=1:NEXT(299$);


;
;
;     Model statements for module:  BasicProcess.Assign 153 (Assign location as
doctors room)
;
299$          ASSIGN:        Track(RFID,2)=7:NEXT(1$);


;
;
;     Model statements for module:  AdvancedProcess.Hold 9 (Doctors wait for a
call)
;
1$            QUEUE,         Doctors wait for a call.Queue;
              SCAN:
ConsultationRoom1NeedDoc+ConsultationRoom2NeedDoc>0:NEXT(266$);


;
;
;     Model statements for module:  BasicProcess.Assign 125 (Assign location as
corridor 5)
;
266$          ASSIGN:        Track(rfid,2)=1:NEXT(103$);


;
;
;     Model statements for module:  BasicProcess.Decide 49 (Choose the
consultation room to go)
```

```
;
103$          BRANCH,        1:
                             If,
                             ConsultationRoom1NeedDoc && (
ConsultationRoom1DoctorRequestTime<=ConsultationRoom2DoctorRequestTime || (1-
ConsultationRoom2NeedDoc)),
                             22$,Yes:
                             If,
                             ConsultationRoom2NeedDoc && (
ConsultationRoom2DoctorRequestTime<=ConsultationRoom1DoctorRequestTime || (1-
ConsultationRoom1NeedDoc)),
                             23$,Yes:
                             Else,103$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 67 (Consultation room 1
doctor call recognized)
;
22$           ASSIGN:        ConsultationRoom1NeedDoc=0:NEXT(115$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 1 (Station 1)
;

115$          STATION,       Station DRCR11;
310$          DELAY:         0.0,,VA:NEXT(117$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 2 (Route 2)
;
117$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station DRCR12;


;
;
;     Model statements for module:  BasicProcess.Assign 68 (Consultation room 2
doctor call recognized)
;
23$           ASSIGN:        ConsultationRoom2NeedDoc=0:NEXT(121$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 5 (Station 5)
;

121$          STATION,       Station DRCR21;
313$          DELAY:         0.0,,VA:NEXT(122$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 4 (Route 4)
;
122$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station DRCR22;


;
```

```
;
;     Model statements for module:  BasicProcess.Create 6 (Create Nurse)
;

314$          CREATE,
6,HoursToBaseTime(0.0),Nurse:HoursToBaseTime(1),1:NEXT(315$);

315$          ASSIGN:        Create Nurse.NumberOut=Create Nurse.NumberOut +
1:NEXT(3$);


;
;
;     Model statements for module:  BasicProcess.Assign 59 (Assign nurse ID)
;
3$            ASSIGN:        NurseCounter=NurseCounter+1:
                             RFID=120+NurseCounter:NEXT(240$);


;
;
;     Model statements for module:  BasicProcess.Assign 103 (Make tag status in
use for nurses)
;
240$          ASSIGN:        Track(rfid,1)=1:NEXT(300$);


;
;
;     Model statements for module:  BasicProcess.Assign 154 (Assign location as
nurses room)
;
300$          ASSIGN:        Track(rfid,2)=6:NEXT(4$);


;
;
;     Model statements for module:  AdvancedProcess.Hold 10 (Nurses wait for a
call)
;
4$            QUEUE,         Nurses wait for a call.Queue;
              SCAN:

ConsultationRoom1NeedNurse+ConsultationRoom2NeedNurse+TreatmentRoom1NeedNurse+T
reatmentRoom2NeedNurse+TreatmentRoom3NeedNurse+TreatmentRoom4NeedNurse+Treatmen
tRoom5NeedNurse+TreatmentRoom6NeedNurse>0
                             :NEXT(267$);


;
;
;     Model statements for module:  BasicProcess.Assign 126 (Assign location as
corridor 6)
;
267$          ASSIGN:        Track(rfid,2)=1:NEXT(94$);


;
;
;     Model statements for module:  BasicProcess.Decide 48 (Choose the room to
go)
;
94$           BRANCH,        1:
```

```
                              If,
                              ConsultationRoom1NeedNurse &&
((ConsultationRoom1NurseRequestTime <= ConsultationRoom2NurseRequestTime) ||
(1-ConsultationRoom2NeedNurse)) && ((ConsultationRoom1NurseRequestTime <=
TreatmentRoom1NurseRequestTime) || (1-TreatmentRoom1NeedNurse)) &&
((ConsultationRoom1NurseRequestTime <= TreatmentRoom2NurseRequestTime) || (1-
TreatmentRoom2NeedNurse)) && ((ConsultationRoom1NurseRequestTime <=
TreatmentRoom3NurseRequestTime) || (1-TreatmentRoom3NeedNurse)) &&
((ConsultationRoom1NurseRequestTime <= TreatmentRoom4NurseRequestTime) || (1-
TreatmentRoom4NeedNurse)) && ((ConsultationRoom1NurseRequestTime <=
TreatmentRoom5NurseRequestTime) || (1-TreatmentRoom5NeedNurse)) &&
((ConsultationRoom1NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                              95$,Yes:
                              If,
                              ConsultationRoom2NeedNurse &&
((ConsultationRoom2NurseRequestTime <= ConsultationRoom1NurseRequestTime) ||
(1-ConsultationRoom1NeedNurse)) && ((ConsultationRoom2NurseRequestTime <=
TreatmentRoom1NurseRequestTime) || (1-TreatmentRoom1NeedNurse)) &&
((ConsultationRoom2NurseRequestTime <= TreatmentRoom2NurseRequestTime) || (1-
TreatmentRoom2NeedNurse)) && ((ConsultationRoom2NurseRequestTime <=
TreatmentRoom3NurseRequestTime) || (1-TreatmentRoom3NeedNurse)) &&
((ConsultationRoom2NurseRequestTime <= TreatmentRoom4NurseRequestTime) || (1-
TreatmentRoom4NeedNurse)) && ((ConsultationRoom2NurseRequestTime <=
TreatmentRoom5NurseRequestTime) || (1-TreatmentRoom5NeedNurse)) &&
((ConsultationRoom2NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                              96$,Yes:
                              If,
                              TreatmentRoom1NeedNurse &&
((TreatmentRoom1NurseRequestTime <= ConsultationRoom1NurseRequestTime) || (1-
ConsultationRoom1NeedNurse)) && ((TreatmentRoom1NurseRequestTime <=
ConsultationRoom2NurseRequestTime) || (1-ConsultationRoom2NeedNurse)) &&
((TreatmentRoom1NurseRequestTime <= TreatmentRoom2NurseRequestTime) || (1-
TreatmentRoom2NeedNurse)) && ((TreatmentRoom1NurseRequestTime <=
TreatmentRoom3NurseRequestTime) || (1-TreatmentRoom3NeedNurse)) &&
((TreatmentRoom1NurseRequestTime <= TreatmentRoom4NurseRequestTime) || (1-
TreatmentRoom4NeedNurse)) && ((TreatmentRoom1NurseRequestTime <=
TreatmentRoom5NurseRequestTime) || (1-TreatmentRoom5NeedNurse)) &&
((TreatmentRoom1NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                              97$,Yes:
                              If,
                              TreatmentRoom2NeedNurse &&
((TreatmentRoom2NurseRequestTime <= ConsultationRoom1NurseRequestTime) || (1-
ConsultationRoom1NeedNurse)) && ((TreatmentRoom2NurseRequestTime <=
ConsultationRoom2NurseRequestTime) || (1-ConsultationRoom2NeedNurse)) &&
((TreatmentRoom2NurseRequestTime <= TreatmentRoom1NurseRequestTime) || (1-
TreatmentRoom1NeedNurse)) && ((TreatmentRoom2NurseRequestTime <=
TreatmentRoom3NurseRequestTime) || (1-TreatmentRoom3NeedNurse)) &&
((TreatmentRoom2NurseRequestTime <= TreatmentRoom4NurseRequestTime) || (1-
TreatmentRoom4NeedNurse)) && ((TreatmentRoom2NurseRequestTime <=
TreatmentRoom5NurseRequestTime) || (1-TreatmentRoom5NeedNurse)) &&
((TreatmentRoom2NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                              98$,Yes:
                              If,
                              TreatmentRoom3NeedNurse &&
((TreatmentRoom3NurseRequestTime <= ConsultationRoom1NurseRequestTime) || (1-
ConsultationRoom1NeedNurse)) && ((TreatmentRoom3NurseRequestTime <=
ConsultationRoom2NurseRequestTime) || (1-ConsultationRoom2NeedNurse)) &&
((TreatmentRoom3NurseRequestTime <= TreatmentRoom1NurseRequestTime) || (1-
TreatmentRoom1NeedNurse)) && ((TreatmentRoom3NurseRequestTime <=
TreatmentRoom2NurseRequestTime) || (1-TreatmentRoom2NeedNurse)) &&
((TreatmentRoom3NurseRequestTime <= TreatmentRoom4NurseRequestTime) || (1-
TreatmentRoom4NeedNurse)) && ((TreatmentRoom3NurseRequestTime <=
TreatmentRoom5NurseRequestTime) || (1-TreatmentRoom5NeedNurse)) &&
((TreatmentRoom3NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                             99$,Yes:
                             If,
                             TreatmentRoom4NeedNurse &&
((TreatmentRoom4NurseRequestTime <= ConsultationRoom1NurseRequestTime) || (1-
ConsultationRoom1NeedNurse)) && ((TreatmentRoom4NurseRequestTime <=
ConsultationRoom2NurseRequestTime) || (1-ConsultationRoom2NeedNurse)) &&
((TreatmentRoom4NurseRequestTime <= TreatmentRoom1NurseRequestTime) || (1-
TreatmentRoom1NeedNurse)) && ((TreatmentRoom4NurseRequestTime <=
TreatmentRoom2NurseRequestTime) || (1-TreatmentRoom2NeedNurse)) &&
((TreatmentRoom4NurseRequestTime <= TreatmentRoom3NurseRequestTime) || (1-
TreatmentRoom3NeedNurse)) && ((TreatmentRoom4NurseRequestTime <=
TreatmentRoom5NurseRequestTime) || (1-TreatmentRoom5NeedNurse)) &&
((TreatmentRoom4NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                             100$,Yes:
                             If,
                             TreatmentRoom5NeedNurse &&
((TreatmentRoom5NurseRequestTime <= ConsultationRoom1NurseRequestTime) || (1-
ConsultationRoom1NeedNurse)) && ((TreatmentRoom5NurseRequestTime <=
ConsultationRoom2NurseRequestTime) || (1-ConsultationRoom2NeedNurse)) &&
((TreatmentRoom5NurseRequestTime <= TreatmentRoom1NurseRequestTime) || (1-
TreatmentRoom1NeedNurse)) && ((TreatmentRoom5NurseRequestTime <=
TreatmentRoom2NurseRequestTime) || (1-TreatmentRoom2NeedNurse)) &&
((TreatmentRoom5NurseRequestTime <= TreatmentRoom3NurseRequestTime) || (1-
TreatmentRoom3NeedNurse)) && ((TreatmentRoom5NurseRequestTime <=
TreatmentRoom4NurseRequestTime) || (1-TreatmentRoom4NeedNurse)) &&
((TreatmentRoom5NurseRequestTime <= TreatmentRoom6NurseRequestTime) || (1-
TreatmentRoom6NeedNurse)),
                             101$,Yes:
                             If,
                             TreatmentRoom6NeedNurse &&
((TreatmentRoom6NurseRequestTime <= ConsultationRoom1NurseRequestTime) || (1-
ConsultationRoom1NeedNurse)) && ((TreatmentRoom6NurseRequestTime <=
ConsultationRoom2NurseRequestTime) || (1-ConsultationRoom2NeedNurse)) &&
((TreatmentRoom6NurseRequestTime <= TreatmentRoom1NurseRequestTime) || (1-
TreatmentRoom1NeedNurse)) && ((TreatmentRoom6NurseRequestTime <=
TreatmentRoom2NurseRequestTime) || (1-TreatmentRoom2NeedNurse)) &&
((TreatmentRoom6NurseRequestTime <= TreatmentRoom3NurseRequestTime) || (1-
TreatmentRoom3NeedNurse)) && ((TreatmentRoom6NurseRequestTime <=
TreatmentRoom4NurseRequestTime) || (1-TreatmentRoom4NeedNurse)) &&
((TreatmentRoom6NurseRequestTime <= TreatmentRoom5NurseRequestTime) || (1-
TreatmentRoom5NeedNurse)),
                             102$,Yes:
                             Else,94$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 84 (Consultation room 1
nurse call recognized)
;
95$           ASSIGN:        ConsultationRoom1NeedNurse=0:NEXT(118$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 3 (Station 3)
;
```

```
118$           STATION,       Station NRCR11;
322$           DELAY:         0.0,,VA:NEXT(119$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 3 (Route 3)
;
119$           ROUTE:         MinutesToBaseTime(NORM( 1.5 , 0.15 )),Station
NRCR12;


;
;
;     Model statements for module:  BasicProcess.Assign 85 (Consultation room 2
nurse call recognized)
;
96$            ASSIGN:        ConsultationRoom2NeedNurse=0:NEXT(124$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 7 (Station 7)
;

124$           STATION,       Station NRCR21;
325$           DELAY:         0.0,,VA:NEXT(125$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 5 (Route 5)
;
125$           ROUTE:         MinutesToBaseTime(NORM( 1.5 , 0.15 )),Station
NRCR22;


;
;
;     Model statements for module:  BasicProcess.Assign 86 (Treatment room 1
nurse call recognized)
;
97$            ASSIGN:        TreatmentRoom1NeedNurse=0:NEXT(167$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 35 (Station 35)
;

167$           STATION,       Station NRTR1;
328$           DELAY:         0.0,,VA:NEXT(168$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 19 (Route 19)
;
168$           ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR1;


;
```

```
;
;     Model statements for module:  BasicProcess.Assign 87 (Treatment room 2
nurse call recognized)
;
98$           ASSIGN:        TreatmentRoom2NeedNurse=0:NEXT(169$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 36 (Station 36)
;

169$          STATION,       Station NRTR2;
331$          DELAY:         0.0,,VA:NEXT(170$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 20 (Route 20)
;
170$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR2;


;
;
;     Model statements for module:  BasicProcess.Assign 88 (Treatment room 3
nurse call recognized)
;
99$           ASSIGN:        TreatmentRoom3NeedNurse=0:NEXT(171$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 37 (Station 37)
;

171$          STATION,       Station NRTR3;
334$          DELAY:         0.0,,VA:NEXT(172$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 21 (Route 21)
;
172$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR3;


;
;
;     Model statements for module:  BasicProcess.Assign 89 (Treatment room 4
nurse call recognized)
;
100$          ASSIGN:        TreatmentRoom4NeedNurse=0:NEXT(173$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 38 (Station 38)
;

173$          STATION,       Station NRTR4;
337$          DELAY:         0.0,,VA:NEXT(174$);
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Route 22 (Route 22)
;
174$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR4;


;
;
;     Model statements for module:  BasicProcess.Assign 90 (Treatment room 5
nurse call recognized)
;
101$          ASSIGN:        TreatmentRoom5NeedNurse=0:NEXT(175$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 39 (Station 39)
;

175$          STATION,       Station NRTR5;
340$          DELAY:         0.0,,VA:NEXT(176$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 23 (Route 23)
;
176$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR5;


;
;
;     Model statements for module:  BasicProcess.Assign 91 (Treatment room 6
nurse call recognized)
;
102$          ASSIGN:        TreatmentRoom6NeedNurse=0:NEXT(177$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 40 (Station 40)
;

177$          STATION,       Station NRTR6;
343$          DELAY:         0.0,,VA:NEXT(178$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 24 (Route 24)
;
178$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR6;


;
;
;     Model statements for module:  BasicProcess.Create 7 (Create Equipment)
;

344$          CREATE,
20,HoursToBaseTime(0.0),Equipment:HoursToBaseTime(1),1:NEXT(345$);
```

```
345$          ASSIGN:        Create Equipment.NumberOut=Create
Equipment.NumberOut + 1:NEXT(6$);


;
;
;     Model statements for module:  BasicProcess.Assign 60 (Assign Equipment
ID)
;
6$            ASSIGN:        EquipmentCounter=EquipmentCounter+1:
                             RFID=100+EquipmentCounter:NEXT(241$);


;
;
;     Model statements for module:  BasicProcess.Assign 104 (Make tag status in
use for equipment)
;
241$          ASSIGN:        Track(rfid,1)=1:NEXT(301$);


;
;
;     Model statements for module:  BasicProcess.Assign 155 (Assign location as
equipment room)
;
301$          ASSIGN:        Track(rfid,2)=14:NEXT(7$);


;
;
;     Model statements for module:  AdvancedProcess.Hold 11 (Equipment wait for
a call)
;
7$            QUEUE,         Equipment wait for a call.Queue;
              SCAN:

TreatmentRoom1NeedEquipment+TreatmentRoom2NeedEquipment+TreatmentRoom3NeedEquip
ment+TreatmentRoom4NeedEquipment+TreatmentRoom5NeedEquipment+TreatmentRoom6Need
Equipment>0
                             :NEXT(287$);


;
;
;     Model statements for module:  BasicProcess.Assign 146 (Assign location as
corridor 25)
;
287$          ASSIGN:        Track(rfid,2)=1:NEXT(104$);


;
;
;     Model statements for module:  BasicProcess.Decide 50 (Choose the
treatment room to go)
;
104$          BRANCH,        1:
                             If,
                             TreatmentRoom1NeedEquipment &&
((TreatmentRoom1EquipmentRequestTime <= TreatmentRoom2EquipmentRequestTime) ||
(1-TreatmentRoom2NeedEquipment)) && ((TreatmentRoom1EquipmentRequestTime <=
TreatmentRoom3EquipmentRequestTime) || (1-TreatmentRoom3NeedEquipment)) &&
((TreatmentRoom1EquipmentRequestTime <= TreatmentRoom4EquipmentRequestTime) ||
(1-TreatmentRoom4NeedEquipment)) && ((TreatmentRoom1EquipmentRequestTime <=
TreatmentRoom5EquipmentRequestTime) || (1-TreatmentRoom5NeedEquipment)) &&
((TreatmentRoom1EquipmentRequestTime <= TreatmentRoom6EquipmentRequestTime) ||
(1-TreatmentRoom6NeedEquipment)),
                              105$,Yes:
                              If,
                              TreatmentRoom2NeedEquipment &&
((TreatmentRoom2EquipmentRequestTime <= TreatmentRoom1EquipmentRequestTime) ||
(1-TreatmentRoom1NeedEquipment)) && ((TreatmentRoom2EquipmentRequestTime <=
TreatmentRoom3EquipmentRequestTime) || (1-TreatmentRoom3NeedEquipment)) &&
((TreatmentRoom2EquipmentRequestTime <= TreatmentRoom4EquipmentRequestTime) ||
(1-TreatmentRoom4NeedEquipment)) && ((TreatmentRoom2EquipmentRequestTime <=
TreatmentRoom5EquipmentRequestTime) || (1-TreatmentRoom5NeedEquipment)) &&
((TreatmentRoom2EquipmentRequestTime <= TreatmentRoom6EquipmentRequestTime) ||
(1-TreatmentRoom6NeedEquipment)),
                              106$,Yes:
                              If,
                              TreatmentRoom3NeedEquipment &&
((TreatmentRoom3EquipmentRequestTime <= TreatmentRoom1EquipmentRequestTime) ||
(1-TreatmentRoom1NeedEquipment)) && ((TreatmentRoom3EquipmentRequestTime <=
TreatmentRoom2EquipmentRequestTime) || (1-TreatmentRoom2NeedEquipment)) &&
((TreatmentRoom3EquipmentRequestTime <= TreatmentRoom4EquipmentRequestTime) ||
(1-TreatmentRoom4NeedEquipment)) && ((TreatmentRoom3EquipmentRequestTime <=
TreatmentRoom5EquipmentRequestTime) || (1-TreatmentRoom5NeedEquipment)) &&
((TreatmentRoom3EquipmentRequestTime <= TreatmentRoom6EquipmentRequestTime) ||
(1-TreatmentRoom6NeedEquipment)),
                              107$,Yes:
                              If,
                              TreatmentRoom4NeedEquipment &&
((TreatmentRoom4EquipmentRequestTime <= TreatmentRoom1EquipmentRequestTime) ||
(1-TreatmentRoom1NeedEquipment)) && ((TreatmentRoom4EquipmentRequestTime <=
TreatmentRoom2EquipmentRequestTime) || (1-TreatmentRoom2NeedEquipment)) &&
((TreatmentRoom4EquipmentRequestTime <= TreatmentRoom3EquipmentRequestTime) ||
(1-TreatmentRoom3NeedEquipment)) && ((TreatmentRoom4EquipmentRequestTime <=
TreatmentRoom5EquipmentRequestTime) || (1-TreatmentRoom5NeedEquipment)) &&
((TreatmentRoom4EquipmentRequestTime <= TreatmentRoom6EquipmentRequestTime) ||
(1-TreatmentRoom6NeedEquipment)),
                              108$,Yes:
                              If,
                              TreatmentRoom5NeedEquipment &&
((TreatmentRoom5EquipmentRequestTime <= TreatmentRoom1EquipmentRequestTime) ||
(1-TreatmentRoom1NeedEquipment)) && ((TreatmentRoom5EquipmentRequestTime <=
TreatmentRoom2EquipmentRequestTime) || (1-TreatmentRoom2NeedEquipment)) &&
((TreatmentRoom5EquipmentRequestTime <= TreatmentRoom3EquipmentRequestTime) ||
(1-TreatmentRoom3NeedEquipment)) && ((TreatmentRoom5EquipmentRequestTime <=
TreatmentRoom4EquipmentRequestTime) || (1-TreatmentRoom4NeedEquipment)) &&
((TreatmentRoom5EquipmentRequestTime <= TreatmentRoom6EquipmentRequestTime) ||
(1-TreatmentRoom6NeedEquipment)),
                              109$,Yes:
                              If,
                              TreatmentRoom6NeedEquipment &&
((TreatmentRoom6EquipmentRequestTime <= TreatmentRoom1EquipmentRequestTime) ||
(1-TreatmentRoom1NeedEquipment)) && ((TreatmentRoom6EquipmentRequestTime <=
TreatmentRoom2EquipmentRequestTime) || (1-TreatmentRoom2NeedEquipment)) &&
((TreatmentRoom6EquipmentRequestTime <= TreatmentRoom3EquipmentRequestTime) ||
(1-TreatmentRoom3NeedEquipment)) && ((TreatmentRoom6EquipmentRequestTime <=
TreatmentRoom4EquipmentRequestTime) || (1-TreatmentRoom4NeedEquipment)) &&
((TreatmentRoom6EquipmentRequestTime <= TreatmentRoom5EquipmentRequestTime) ||
(1-TreatmentRoom5NeedEquipment)),
                              110$,Yes:
                              Else,104$,Yes;
```

```
;
;
;     Model statements for module:  BasicProcess.Assign 92 (Treatment room 1
equipment call recognized)
;
105$          ASSIGN:        TreatmentRoom1NeedEquipment=0:NEXT(179$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 41 (Station 41)
;

179$          STATION,       Station ERTR1;
352$          DELAY:         0.0,,VA:NEXT(180$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 25 (Route 25)
;
180$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR1;


;
;
;     Model statements for module:  BasicProcess.Assign 93 (Treatment room 2
equipment call recognized)
;
106$          ASSIGN:        TreatmentRoom2NeedEquipment=0:NEXT(181$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 42 (Station 42)
;

181$          STATION,       Station ERTR2;
355$          DELAY:         0.0,,VA:NEXT(182$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 26 (Route 26)
;
182$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR2;


;
;
;     Model statements for module:  BasicProcess.Assign 94 (Treatment room 3
equipment call recognized)
;
107$          ASSIGN:        TreatmentRoom3NeedEquipment=0:NEXT(183$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 43 (Station 43)
;

183$          STATION,       Station ERTR3;
358$          DELAY:         0.0,,VA:NEXT(184$);
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Route 27 (Route 27)
;
184$          ROUTE:          MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR3;


;
;
;     Model statements for module:  BasicProcess.Assign 95 (Treatment room 4
equipment call recognized)
;
108$          ASSIGN:         TreatmentRoom4NeedEquipment=0:NEXT(185$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 44 (Station 44)
;

185$          STATION,        Station ERTR4;
361$          DELAY:          0.0,,VA:NEXT(186$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 28 (Route 28)
;
186$          ROUTE:          MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR4;


;
;
;     Model statements for module:  BasicProcess.Assign 96 (Treatment room 5
equipment call recognized)
;
109$          ASSIGN:         TreatmentRoom5NeedEquipment=0:NEXT(187$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 45 (Station 45)
;

187$          STATION,        Station ERTR5;
364$          DELAY:          0.0,,VA:NEXT(188$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 29 (Route 29)
;
188$          ROUTE:          MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR5;


;
;
;     Model statements for module:  BasicProcess.Assign 97 (Treatment room 6
equipment call recognized)
;
110$          ASSIGN:         TreatmentRoom6NeedEquipment=0:NEXT(189$);
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Station 46 (Station 46)
;

189$          STATION,       Station ERTR6;
367$          DELAY:         0.0,,VA:NEXT(190$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 30 (Route 30)
;
190$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station TR6;


;
;
;     Model statements for module:  BasicProcess.Create 8 (Create Patient)
;

368$          CREATE,
1,MinutesToBaseTime(0.01),Patient:MinutesToBaseTime(EXPO(8)):NEXT(369$);

369$          ASSIGN:        Create Patient.NumberOut=Create Patient.NumberOut
+ 1:NEXT(9$);


;
;
;     Model statements for module:  BasicProcess.Assign 61 (Assign PatientID)
;
9$            ASSIGN:        PatientCounter=PatientCounter+1:
                             PatientID=PatientCounter:NEXT(10$);


;
;
;     Model statements for module:  BasicProcess.Process 21 (Register 1)
;
10$           ASSIGN:        Register 1.NumberIn=Register 1.NumberIn + 1:
                             Register 1.WIP=Register 1.WIP+1;
375$          QUEUE,         Register 1.Queue;
374$          SEIZE,         2,VA:
                             registrar,1:NEXT(373$);

373$          DELAY:         MinutesToBaseTime(Normal(2,0.2)),,VA;
420$          ASSIGN:        Register 1.NumberOut=Register 1.NumberOut + 1:
                             Register 1.WIP=Register 1.WIP-1:NEXT(11$);


;
;
;     Model statements for module:  BasicProcess.Decide 36 (Choose Tag)
;
11$           BRANCH,        1:
                             If,Track(RFIDCounter,1)==0,423$,Yes:
                             Else,424$,Yes;
423$          ASSIGN:        Choose Tag.NumberOut True=Choose Tag.NumberOut
True + 1:NEXT(13$);
```

```
424$          ASSIGN:        Choose Tag.NumberOut False=Choose Tag.NumberOut
False + 1:NEXT(12$);


;
;
;     Model statements for module:  BasicProcess.Assign 63 (Assign Tag)
;
13$           ASSIGN:        RFID=RFIDCounter:
                             RFIDCounter=1:NEXT(14$);


;
;
;     Model statements for module:  BasicProcess.Assign 64 (Make tag status in
use for patients)
;
14$           ASSIGN:        Track(rfid,1)=1:NEXT(298$);


;
;
;     Model statements for module:  BasicProcess.Assign 152 (Assign location as
register 1)
;
298$          ASSIGN:        Track(rfid,2)=2:NEXT(15$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 23 (Write to
visit table)
;
15$           WRITE,         File 1,RECORDSET(Recordset 2):
                             PatientID,
                             RFID:NEXT(16$);


;
;
;     Model statements for module:  BasicProcess.Process 22 (Register 2)
;
16$           ASSIGN:        Register 2.NumberIn=Register 2.NumberIn + 1:
                             Register 2.WIP=Register 2.WIP+1;
426$          DELAY:         MinutesToBaseTime(Normal(2,.2)),,VA;
425$          RELEASE:       registrar,1;
473$          ASSIGN:        Register 2.NumberOut=Register 2.NumberOut + 1:
                             Register 2.WIP=Register 2.WIP-1:NEXT(263$);


;
;
;     Model statements for module:  BasicProcess.Assign 122 (Assign location as
corridor 1)
;
263$          ASSIGN:        Track(rfid,2)=1:NEXT(133$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 13 (Station 13)
;
```

```
133$          STATION,       Station RWR1;
478$          DELAY:         0.0,,VA:NEXT(135$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 8 (Route 8)
;
135$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station RWR2;


;
;
;     Model statements for module:  BasicProcess.Assign 62 (Browse Tag)
;
12$           ASSIGN:        RFIDCounter=rfidcounter+1:NEXT(11$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 2 (Station 2)
;

116$          STATION,       Station DRCR12;
481$          DELAY:         0.0,,VA:NEXT(243$);


;
;
;     Model statements for module:  BasicProcess.Assign 106 (Assign location as
consultation room 1)
;
243$          ASSIGN:        Track(rfid,2)=4:NEXT(24$);


;
;
;     Model statements for module:  BasicProcess.Batch 19 (Wait for others
before consultation 1)
;
24$           QUEUE,         Wait for others before consultation 1.Queue;
482$          GROUP,         ,Temporary:3,Last,Patient:NEXT(483$);

483$          ASSIGN:        Wait for others before consultation
1.NumberOut=Wait for others before consultation 1.NumberOut + 1
                             :NEXT(26$);


;
;
;     Model statements for module:  BasicProcess.Process 23 (Consultation 1)
;
26$           ASSIGN:        Consultation 1.NumberIn=Consultation 1.NumberIn +
1:
                             Consultation 1.WIP=Consultation 1.WIP+1;
487$          QUEUE,         Consultation 1.Queue;
486$          SEIZE,         2,VA:
                             CR1,1:NEXT(485$);

485$          DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
484$          RELEASE:       CR1,1;
532$          ASSIGN:        Consultation 1.NumberOut=Consultation 1.NumberOut
+ 1:
```

```
                             Consultation 1.WIP=Consultation 1.WIP-1:NEXT(28$);


;
;
;     Model statements for module:  BasicProcess.Assign 69 (Consultation Room1
Empty)
;
28$           ASSIGN:        ConsultationRoom1Status=0:NEXT(30$);


;
;
;     Model statements for module:  BasicProcess.Separate 15 (Separate after
consultation 1)
;
30$           SPLIT::NEXT(535$);

535$          ASSIGN:        Separate after consultation 1.NumberOut
Orig=Separate after consultation 1.NumberOut Orig + 1
                             :NEXT(268$);


;
;
;     Model statements for module:  BasicProcess.Assign 127 (Assign location as
corridor 4)
;
268$          ASSIGN:        Track(rfid,2)=1:NEXT(32$);


;
;
;     Model statements for module:  BasicProcess.Decide 38 (Route entities 1)
;
32$           BRANCH,        1:
                             If,Entity.Type==Doctor,127$,Yes:
                             If,Entity.Type==Patient,33$,Yes:
                             If,Entity.Type==Nurse,130$,Yes:
                             Else,32$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 9 (Station 9)
;

127$          STATION,       Station CRDR1;
542$          DELAY:         0.0,,VA:NEXT(128$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 6 (Route 6)
;
128$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station CRDR2;


;
;
;     Model statements for module:  BasicProcess.Decide 39 (Go to treatment or
leave?)
;
33$           BRANCH,        1:
```

```
                           With,(75)/100,543$,Yes:
                           Else,544$,Yes;
543$          ASSIGN:       Go to treatment or leave?.NumberOut True=Go to
treatment or leave?.NumberOut True + 1:NEXT(146$);

544$          ASSIGN:       Go to treatment or leave?.NumberOut False=Go to
treatment or leave?.NumberOut False + 1:NEXT(142$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 21 (Station 21)
;

146$          STATION,      Station CRWR1;
547$          DELAY:        0.0,,VA:NEXT(147$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 12 (Route 12)
;
147$          ROUTE:        MinutesToBaseTime(NORM( 1 , 0.1 )),Station CRWR2;


;
;
;     Model statements for module:  AdvancedTransfer.Station 19 (Station 19)
;

142$          STATION,      Station CRR1;
550$          DELAY:        0.0,,VA:NEXT(143$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 11 (Route 11)
;
143$          ROUTE:        MinutesToBaseTime(NORM( 1 , 0.1 )),Station CRR2;


;
;
;     Model statements for module:  AdvancedTransfer.Station 11 (Station 11)
;

130$          STATION,      Station CRNR1;
553$          DELAY:        0.0,,VA:NEXT(131$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 7 (Route 7)
;
131$          ROUTE:        MinutesToBaseTime(NORM( 1 , 0.1 )),Station CRNR2;


;
;
;     Model statements for module:  AdvancedTransfer.Station 4 (Station 4)
;

120$          STATION,      Station NRCR12;
```

```
556$          DELAY:         0.0,,VA:NEXT(243$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 6 (Station 6)
;

123$          STATION,       Station DRCR22;
559$          DELAY:         0.0,,VA:NEXT(244$);


;
;
;     Model statements for module:  BasicProcess.Assign 107 (Assign location as
consultation room 2)
;
244$          ASSIGN:        Track(rfid,2)=5:NEXT(25$);


;
;
;     Model statements for module:  BasicProcess.Batch 20 (Wait for others
before consultation 2)
;
25$           QUEUE,         Wait for others before consultation 2.Queue;
560$          GROUP,         ,Temporary:3,Last,Patient:NEXT(561$);

561$          ASSIGN:        Wait for others before consultation
2.NumberOut=Wait for others before consultation 2.NumberOut + 1
                             :NEXT(27$);


;
;
;     Model statements for module:  BasicProcess.Process 24 (Consultation 2)
;
27$           ASSIGN:        Consultation 2.NumberIn=Consultation 2.NumberIn +
1:
                             Consultation 2.WIP=Consultation 2.WIP+1;
565$          QUEUE,         Consultation 2.Queue;
564$          SEIZE,         2,VA:
                             CR2,1:NEXT(563$);

563$          DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
562$          RELEASE:       CR2,1;
610$          ASSIGN:        Consultation 2.NumberOut=Consultation 2.NumberOut
+ 1:
                             Consultation 2.WIP=Consultation 2.WIP-1:NEXT(29$);


;
;
;     Model statements for module:  BasicProcess.Assign 70 (Consultation Room2
Empty)
;
29$           ASSIGN:        ConsultationRoom2Status=0:NEXT(31$);


;
;
;     Model statements for module:  BasicProcess.Separate 16 (Separate after
consultation 2)
```

```
;
31$           SPLIT::NEXT(613$);

613$          ASSIGN:        Separate after consultation 2.NumberOut
Orig=Separate after consultation 2.NumberOut Orig + 1
                             :NEXT(268$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 8 (Station 8)
;

126$          STATION,       Station NRCR22;
618$          DELAY:         0.0,,VA:NEXT(244$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 10 (Station 10)
;

129$          STATION,       Station CRDR2;
621$          DELAY:         0.0,,VA:NEXT(239$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 12 (Station 12)
;

132$          STATION,       Station CRNR2;
624$          DELAY:         0.0,,VA:NEXT(240$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 14 (Station 14)
;

134$          STATION,       Station RWR2;
627$          DELAY:         0.0,,VA:NEXT(242$);


;
;
;     Model statements for module:  BasicProcess.Assign 105 (Assign location as
waiting room 1)
;
242$          ASSIGN:        Track(rfid,2)=3:NEXT(19$);


;
;
;     Model statements for module:  BasicProcess.Decide 37 (Check consultation
room availability)
;
19$           BRANCH,        1:
                             If,ConsultationRoom1Status==0,20$,Yes:
                             If,ConsultationRoom2Status==0,21$,Yes:
                             Else,17$,Yes;

;
```

```
;
;     Model statements for module:  AdvancedProcess.Hold 13 (Wait for available
consultation room)
;
17$           QUEUE,         Wait for available consultation room.Queue;
              SCAN:          ConsultationRoom1Status + ConsultationRoom2Status
<2:NEXT(19$);


;
;
;     Model statements for module:  BasicProcess.Assign 65 (Consultation Room1
Full)
;
20$           ASSIGN:        ConsultationRoom1NurseRequestTime=tnow:
                             ConsultationRoom1DoctorRequestTime=tnow:
                             ConsultationRoom1NeedDoc=1:
                             ConsultationRoom1NeedNurse=1:
                             ConsultationRoom1Status=1:NEXT(264$);


;
;
;     Model statements for module:  BasicProcess.Assign 123 (Assign location as
corridor 2)
;
264$          ASSIGN:        Track(rfid,2)=1:NEXT(136$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 15 (Station 15)
;

136$          STATION,       Station WRCR11;
632$          DELAY:         0.0,,VA:NEXT(137$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 9 (Route 9)
;
137$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station WRCR12;


;
;
;     Model statements for module:  BasicProcess.Assign 66 (Consultation Room 2
Full)
;
21$           ASSIGN:        ConsultationRoom2DoctorRequestTime=tnow:
                             ConsultationRoom2NurseRequestTime=tnow:
                             ConsultationRoom2NeedDoc=1:
                             ConsultationRoom2NeedNurse=1:
                             ConsultationRoom2Status=1:NEXT(265$);


;
;
;     Model statements for module:  BasicProcess.Assign 124 (Assign location as
corridor 3)
;
265$          ASSIGN:        Track(rfid,2)=1:NEXT(139$);
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Station 17 (Station 17)
;

139$          STATION,       Station WRCR21;
635$          DELAY:         0.0,,VA:NEXT(140$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 10 (Route 10)
;
140$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station WRCR22;


;
;
;     Model statements for module:  AdvancedTransfer.Station 16 (Station 16)
;

138$          STATION,       Station WRCR12;
638$          DELAY:         0.0,,VA:NEXT(243$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 18 (Station 18)
;

141$          STATION,       Station WRCR22;
641$          DELAY:         0.0,,VA:NEXT(244$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 20 (Station 20)
;

144$          STATION,       Station CRR2;
644$          DELAY:         0.0,,VA:NEXT(245$);


;
;
;     Model statements for module:  BasicProcess.Assign 108 (Assign location as
register 2)
;
245$          ASSIGN:        Track(rfid,2)=2:NEXT(145$);


;
;
;     Model statements for module:  BasicProcess.Process 40 (Register 3)
;
145$          ASSIGN:        Register 3.NumberIn=Register 3.NumberIn + 1:
                             Register 3.WIP=Register 3.WIP+1;
648$          QUEUE,         Register 3.Queue;
647$          SEIZE,         2,VA:
                             registrar,1:NEXT(646$);
```

```
646$          DELAY:         MinutesToBaseTime(Normal(1,0.1)),,VA;
645$          RELEASE:       registrar,1;
693$          ASSIGN:        Register 3.NumberOut=Register 3.NumberOut + 1:
                             Register 3.WIP=Register 3.WIP-1:NEXT(35$);


;
;
;     Model statements for module:  BasicProcess.Assign 71 (Release Tag 1)
;
35$           ASSIGN:        Track(rfid,1)=0:NEXT(34$);


;
;
;     Model statements for module:  BasicProcess.Dispose 10 (Departure after
consultation)
;
34$           ASSIGN:        Departure after consultation.NumberOut=Departure
after consultation.NumberOut + 1;
696$          DISPOSE:       Yes;


;
;
;     Model statements for module:  AdvancedTransfer.Station 22 (Station 22)
;

148$          STATION,       Station CRWR2;
699$          DELAY:         0.0,,VA:NEXT(246$);


;
;
;     Model statements for module:  BasicProcess.Assign 109 (Assign location as
waiting room 2)
;
246$          ASSIGN:        Track(rfid,2)=3:NEXT(38$);


;
;
;     Model statements for module:  BasicProcess.Decide 40 (Check treatment
room availability)
;
38$           BRANCH,        1:
                             If,TreatmentRoom1Status==0,269$,Yes:
                             If,TreatmentRoom2Status==0,270$,Yes:
                             If,TreatmentRoom3Status==0,271$,Yes:
                             If,TreatmentRoom4Status==0,272$,Yes:
                             If,TreatmentRoom5Status==0,273$,Yes:
                             If,TreatmentRoom6Status==0,274$,Yes:
                             Else,36$,Yes;

;
;
;     Model statements for module:  AdvancedProcess.Hold 14 (Wait for available
treatment room)
;
36$           QUEUE,         Wait for available treatment room.Queue;
              SCAN:
```

```
TreatmentRoom1Status+TreatmentRoom2Status+TreatmentRoom3Status+TreatmentRoom4St
atus+TreatmentRoom5Status+TreatmentRoom6Status<6
                              :NEXT(38$);


;
;
;     Model statements for module:  BasicProcess.Assign 128 (Assign location as
corridor 7)
;
269$          ASSIGN:        Track(rfid,2)=1:NEXT(39$);


;
;
;     Model statements for module:  BasicProcess.Assign 72 (Treatment Room1
Full)
;
39$           ASSIGN:        TreatmentRoom1NurseRequestTime=tnow:
                             TreatmentRoom1EquipmentRequestTime=tnow:
                             TreatmentRoom1NeedEquipment=1:
                             TreatmentRoom1NeedNurse=1:
                             TreatmentRoom1Status=1:NEXT(149$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 23 (Station 23)
;

149$          STATION,       Station WRTR1;
704$          DELAY:         0.0,,VA:NEXT(150$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 13 (Route 13)
;
150$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station TR1;


;
;
;     Model statements for module:  BasicProcess.Assign 129 (Assign location as
corridor 8)
;
270$          ASSIGN:        Track(rfid,2)=1:NEXT(40$);


;
;
;     Model statements for module:  BasicProcess.Assign 73 (Treatment Room2
Full)
;
40$           ASSIGN:        TreatmentRoom2NurseRequestTime=tnow:
                             TreatmentRoom2EquipmentRequestTime=tnow:
                             TreatmentRoom2NeedEquipment=1:
                             TreatmentRoom2NeedNurse=1:
                             TreatmentRoom2Status=1:NEXT(152$);


;
```

```
;
;     Model statements for module:  AdvancedTransfer.Station 25 (Station 25)
;

152$          STATION,       Station WRTR2;
707$          DELAY:         0.0,,VA:NEXT(153$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 14 (Route 14)
;
153$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station TR2;


;
;
;     Model statements for module:  BasicProcess.Assign 130 (Assign location as
corridor 9)
;
271$          ASSIGN:        Track(rfid,2)=1:NEXT(41$);


;
;
;     Model statements for module:  BasicProcess.Assign 74 (Treatment Room3
Full)
;
41$           ASSIGN:        TreatmentRoom3NurseRequestTime=tnow:
                             TreatmentRoom3EquipmentRequestTime=tnow:
                             TreatmentRoom3NeedEquipment=1:
                             TreatmentRoom3NeedNurse=1:
                             TreatmentRoom3Status=1:NEXT(155$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 27 (Station 27)
;

155$          STATION,       Station WRTR3;
710$          DELAY:         0.0,,VA:NEXT(156$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 15 (Route 15)
;
156$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station TR3;


;
;
;     Model statements for module:  BasicProcess.Assign 131 (Assign location as
corridor 10)
;
272$          ASSIGN:        Track(rfid,2)=1:NEXT(42$);


;
;
;     Model statements for module:  BasicProcess.Assign 75 (Treatment Room4
Full)
```

```
;
42$           ASSIGN:        TreatmentRoom4NurseRequestTime=tnow:
                             TreatmentRoom4EquipmentRequestTime=tnow:
                             TreatmentRoom4NeedEquipment=1:
                             TreatmentRoom4NeedNurse=1:
                             TreatmentRoom4Status=1:NEXT(158$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 29 (Station 29)
;

158$          STATION,       Station WRTR4;
713$          DELAY:         0.0,,VA:NEXT(159$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 16 (Route 16)
;
159$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station TR4;


;
;
;     Model statements for module:  BasicProcess.Assign 132 (Assign location as
corridor 11)
;
273$          ASSIGN:        Track(rfid,2)=1:NEXT(43$);


;
;
;     Model statements for module:  BasicProcess.Assign 76 (Treatment Room5
Full)
;
43$           ASSIGN:        TreatmentRoom5NurseRequestTime=tnow:
                             TreatmentRoom5EquipmentRequestTime=tnow:
                             TreatmentRoom5NeedEquipment=1:
                             TreatmentRoom5NeedNurse=1:
                             TreatmentRoom5Status=1:NEXT(161$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 31 (Station 31)
;

161$          STATION,       Station WRTR5;
716$          DELAY:         0.0,,VA:NEXT(162$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 17 (Route 17)
;
162$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station TR5;


;
;
```

```
;     Model statements for module:  BasicProcess.Assign 133 (Assign location as
corridor 12)
;
274$          ASSIGN:        Track(rfid,2)=1:NEXT(44$);


;
;
;     Model statements for module:  BasicProcess.Assign 77 (Treatment Room6
Full)
;
44$           ASSIGN:        TreatmentRoom6NurseRequestTime=tnow:
                             TreatmentRoom6EquipmentRequestTime=tnow:
                             TreatmentRoom6NeedEquipment=1:
                             TreatmentRoom6NeedNurse=1:
                             TreatmentRoom6Status=1:NEXT(164$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 33 (Station 33)
;

164$          STATION,       Station WRTR6;
719$          DELAY:         0.0,,VA:NEXT(165$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 18 (Route 18)
;
165$          ROUTE:         MinutesToBaseTime(NORM( 1 , 0.1 )),Station TR6;


;
;
;     Model statements for module:  AdvancedTransfer.Station 24 (Station 24)
;

151$          STATION,       Station TR1;
722$          DELAY:         0.0,,VA:NEXT(247$);


;
;
;     Model statements for module:  BasicProcess.Assign 110 (Assign location as
treatment room 1)
;
247$          ASSIGN:        Track(rfid,2)=8:NEXT(45$);


;
;
;     Model statements for module:  BasicProcess.Batch 21 (Wait for others
before treatment set up 1)
;
45$           QUEUE,         Wait for others before treatment set up 1.Queue;
723$          GROUP,         ,Temporary:3,Last,Patient:NEXT(724$);

724$          ASSIGN:        Wait for others before treatment set up
1.NumberOut=
                             Wait for others before treatment set up
1.NumberOut + 1:NEXT(51$);
```

```
;
;
;     Model statements for module:  BasicProcess.Process 25 (Treatment Set Up
1)
;
51$           ASSIGN:        Treatment Set Up 1.NumberIn=Treatment Set Up
1.NumberIn + 1:
                             Treatment Set Up 1.WIP=Treatment Set Up 1.WIP+1;
728$          QUEUE,         Treatment Set Up 1.Queue;
727$          SEIZE,         2,VA:
                             TSU1,1:NEXT(726$);

726$          DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
725$          RELEASE:       TSU1,1;
773$          ASSIGN:        Treatment Set Up 1.NumberOut=Treatment Set Up
1.NumberOut + 1:
                             Treatment Set Up 1.WIP=Treatment Set Up 1.WIP-
1:NEXT(57$);


;
;
;     Model statements for module:  BasicProcess.Separate 17 (Separate after
treatment set up 1)
;
57$           SPLIT::NEXT(776$);

776$          ASSIGN:        Separate after treatment set up 1.NumberOut Orig=
                             Separate after treatment set up 1.NumberOut Orig +
1:NEXT(63$);


;
;
;     Model statements for module:  BasicProcess.Decide 41 (Reroute entities
after treatment set up 1)
;
63$           BRANCH,        1:
                             If,Entity.Type==Patient,69$,Yes:
                             If,Entity.Type==Equipment,69$,Yes:
                             Else,275$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 134 (Assing location as
corridor 13)
;
275$          ASSIGN:        Track(rfid,2)=1:NEXT(191$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 47 (Station 47)
;

191$          STATION,       Station TR1NR;
783$          DELAY:         0.0,,VA:NEXT(192$);


;
;
```

```
;     Model statements for module:  AdvancedTransfer.Route 31 (Route 31)
;
192$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station NR;


;
;
;     Model statements for module:  BasicProcess.Batch 27 (Wait for other
before treatment 1)
;
69$           QUEUE,         Wait for other before treatment 1.Queue;
784$          GROUP,         ,Temporary:2,Last,Patient:NEXT(785$);

785$          ASSIGN:        Wait for other before treatment 1.NumberOut=Wait
for other before treatment 1.NumberOut + 1
                             :NEXT(75$);


;
;
;     Model statements for module:  BasicProcess.Process 31 (Treatment 1)
;
75$           ASSIGN:        Treatment 1.NumberIn=Treatment 1.NumberIn + 1:
                             Treatment 1.WIP=Treatment 1.WIP+1;
789$          QUEUE,         Treatment 1.Queue;
788$          SEIZE,         2,VA:
                             T1,1:NEXT(787$);

787$          DELAY:         MinutesToBaseTime(Normal(30,3)),,VA;
786$          RELEASE:       T1,1;
834$          ASSIGN:        Treatment 1.NumberOut=Treatment 1.NumberOut + 1:
                             Treatment 1.WIP=Treatment 1.WIP-1:NEXT(81$);


;
;
;     Model statements for module:  BasicProcess.Assign 78 (Treatment Room1
Empty)
;
81$           ASSIGN:        TreatmentRoom1Status=0:NEXT(87$);


;
;
;     Model statements for module:  BasicProcess.Separate 23 (Separate after
treatment 1)
;
87$           SPLIT::NEXT(837$);

837$          ASSIGN:        Separate after treatment 1.NumberOut Orig=Separate
after treatment 1.NumberOut Orig + 1:NEXT(281$);


;
;
;     Model statements for module:  BasicProcess.Assign 140 (Assign location as
corridor 19)
;
281$          ASSIGN:        Track(rfid,2)=1:NEXT(93$);


;
;
```

```
;     Model statements for module:  BasicProcess.Decide 47 (Reroute entities
after treatment 1)
;
93$           BRANCH,        1:
                             If,Entity.Type==Equipment,204$,Yes:
                             If,Entity.Type==Patient,222$,Yes:
                             Else,93$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 54 (Station 54)
;

204$          STATION,       Station TR1ECR;
844$          DELAY:         0.0,,VA:NEXT(205$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 37 (Route 37)
;
205$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station ECR;


;
;
;     Model statements for module:  AdvancedTransfer.Station 61 (Station 61)
;

222$          STATION,       Station TR1R;
847$          DELAY:         0.0,,VA:NEXT(223$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 43 (Route 43)
;
223$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station R;


;
;
;     Model statements for module:  AdvancedTransfer.Station 26 (Station 26)
;

154$          STATION,       Station TR2;
850$          DELAY:         0.0,,VA:NEXT(248$);


;
;
;     Model statements for module:  BasicProcess.Assign 111 (Assign location as
treatment room 2)
;
248$          ASSIGN:        Track(rfid,2)=9:NEXT(46$);


;
;
;     Model statements for module:  BasicProcess.Batch 22 (Wait for others
before treatment set up 2)
;
46$           QUEUE,         Wait for others before treatment set up 2.Queue;
```

```
851$          GROUP,         ,Temporary:3,Last,Patient:NEXT(852$);

852$          ASSIGN:        Wait for others before treatment set up
2.NumberOut=
                             Wait for others before treatment set up
2.NumberOut + 1:NEXT(52$);


;
;
;     Model statements for module:  BasicProcess.Process 26 (Treatment Set Up
2)
;
52$           ASSIGN:        Treatment Set Up 2.NumberIn=Treatment Set Up
2.NumberIn + 1:
                             Treatment Set Up 2.WIP=Treatment Set Up 2.WIP+1;
856$          QUEUE,         Treatment Set Up 2.Queue;
855$          SEIZE,         2,VA:
                             TSU2,1:NEXT(854$);

854$          DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
853$          RELEASE:       TSU2,1;
901$          ASSIGN:        Treatment Set Up 2.NumberOut=Treatment Set Up
2.NumberOut + 1:
                             Treatment Set Up 2.WIP=Treatment Set Up 2.WIP-
1:NEXT(58$);


;
;
;     Model statements for module:  BasicProcess.Separate 18 (Separate after
treatment set up 2)
;
58$           SPLIT::NEXT(904$);

904$          ASSIGN:        Separate after treatment set up 2.NumberOut Orig=
                             Separate after treatment set up 2.NumberOut Orig +
1:NEXT(64$);


;
;
;     Model statements for module:  BasicProcess.Decide 42 (Reroute entities
after treatment set up 2)
;
64$           BRANCH,        1:
                             If,Entity.Type==Patient,70$,Yes:
                             If,Entity.Type==Equipment,70$,Yes:
                             Else,276$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 135 (Assing location as
corridor 14)
;
276$          ASSIGN:        Track(rfid,2)=1:NEXT(194$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 49 (Station 49)
;
```

```
194$          STATION,       Station TR2NR;
911$          DELAY:         0.0,,VA:NEXT(195$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 32 (Route 32)
;
195$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station NR;


;
;
;     Model statements for module:  BasicProcess.Batch 28 (Wait for other
before treatment 2)
;
70$           QUEUE,         Wait for other before treatment 2.Queue;
912$          GROUP,         ,Temporary:2,Last,Patient:NEXT(913$);

913$          ASSIGN:        Wait for other before treatment 2.NumberOut=Wait
for other before treatment 2.NumberOut + 1
                             :NEXT(76$);


;
;
;     Model statements for module:  BasicProcess.Process 32 (Treatment 2)
;
76$           ASSIGN:        Treatment 2.NumberIn=Treatment 2.NumberIn + 1:
                             Treatment 2.WIP=Treatment 2.WIP+1;
917$          QUEUE,         Treatment 2.Queue;
916$          SEIZE,         2,VA:
                             T2,1:NEXT(915$);

915$          DELAY:         MinutesToBaseTime(Normal(30,3)),,VA;
914$          RELEASE:       T2,1;
962$          ASSIGN:        Treatment 2.NumberOut=Treatment 2.NumberOut + 1:
                             Treatment 2.WIP=Treatment 2.WIP-1:NEXT(82$);


;
;
;     Model statements for module:  BasicProcess.Assign 79 (Treatment Room2
Empty)
;
82$           ASSIGN:        TreatmentRoom2Status=0:NEXT(88$);


;
;
;     Model statements for module:  BasicProcess.Separate 24 (Separate after
treatment 2)
;
88$           SPLIT::NEXT(965$);

965$          ASSIGN:        Separate after treatment 2.NumberOut Orig=Separate
after treatment 2.NumberOut Orig + 1:NEXT(282$);


;
;
;     Model statements for module:  BasicProcess.Assign 141 (Assign location as
corridor 20)
```

```
;
282$          ASSIGN:        Track(rfid,2)=1:NEXT(206$);


;
;
;     Model statements for module:  BasicProcess.Decide 51 (Reroute entities
after treatment 2)
;
206$          BRANCH,        1:
                             If,Entity.Type==Equipment,207$,Yes:
                             If,Entity.Type==Patient,226$,Yes:
                             Else,206$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 55 (Station 55)
;

207$          STATION,       Station TR2ECR;
972$          DELAY:         0.0,,VA:NEXT(208$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 38 (Route 38)
;
208$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station ECR;


;
;
;     Model statements for module:  AdvancedTransfer.Station 63 (Station 63)
;

226$          STATION,       Station TR2R;
975$          DELAY:         0.0,,VA:NEXT(227$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 44 (Route 44)
;
227$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station R;


;
;
;     Model statements for module:  AdvancedTransfer.Station 28 (Station 28)
;

157$          STATION,       Station TR3;
978$          DELAY:         0.0,,VA:NEXT(249$);


;
;
;     Model statements for module:  BasicProcess.Assign 112 (Assign location as
treatment room 3)
;
249$          ASSIGN:        Track(rfid,2)=10:NEXT(47$);
```

```
;
;
;     Model statements for module:  BasicProcess.Batch 23 (Wait for others
before treatment set up 3)
;
47$           QUEUE,         Wait for others before treatment set up 3.Queue;
979$          GROUP,         ,Temporary:3,Last,Patient:NEXT(980$);

980$          ASSIGN:        Wait for others before treatment set up
3.NumberOut=
                             Wait for others before treatment set up
3.NumberOut + 1:NEXT(53$);


;
;
;     Model statements for module:  BasicProcess.Process 27 (Treatment Set Up
3)
;
53$           ASSIGN:        Treatment Set Up 3.NumberIn=Treatment Set Up
3.NumberIn + 1:
                             Treatment Set Up 3.WIP=Treatment Set Up 3.WIP+1;
984$          QUEUE,         Treatment Set Up 3.Queue;
983$          SEIZE,         2,VA:
                             TSU3,1:NEXT(982$);

982$          DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
981$          RELEASE:       TSU3,1;
1029$         ASSIGN:        Treatment Set Up 3.NumberOut=Treatment Set Up
3.NumberOut + 1:
                             Treatment Set Up 3.WIP=Treatment Set Up 3.WIP-
1:NEXT(59$);


;
;
;     Model statements for module:  BasicProcess.Separate 19 (Separate after
treatment set up 3)
;
59$           SPLIT::NEXT(1032$);

1032$         ASSIGN:        Separate after treatment set up 3.NumberOut Orig=
                             Separate after treatment set up 3.NumberOut Orig +
1:NEXT(65$);


;
;
;     Model statements for module:  BasicProcess.Decide 43 (Reroute entities
after treatment set up 3)
;
65$           BRANCH,        1:
                             If,Entity.Type==Patient,71$,Yes:
                             If,Entity.Type==Equipment,71$,Yes:
                             Else,277$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 136 (Assing location as
corridor 15)
;
277$          ASSIGN:        Track(rfid,2)=1:NEXT(196$);
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Station 50 (Station 50)
;

196$           STATION,       Station TR3NR;
1039$          DELAY:         0.0,,VA:NEXT(197$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 33 (Route 33)
;
197$           ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station NR;


;
;
;     Model statements for module:  BasicProcess.Batch 29 (Wait for other
before treatment 3)
;
71$            QUEUE,         Wait for other before treatment 3.Queue;
1040$          GROUP,         ,Temporary:2,Last,Patient:NEXT(1041$);

1041$          ASSIGN:        Wait for other before treatment 3.NumberOut=Wait
for other before treatment 3.NumberOut + 1
                              :NEXT(77$);


;
;
;     Model statements for module:  BasicProcess.Process 33 (Treatment 3)
;
77$            ASSIGN:        Treatment 3.NumberIn=Treatment 3.NumberIn + 1:
                              Treatment 3.WIP=Treatment 3.WIP+1;
1045$          QUEUE,         Treatment 3.Queue;
1044$          SEIZE,         2,VA:
                              T3,1:NEXT(1043$);

1043$          DELAY:         MinutesToBaseTime(Normal(30,3)),,VA;
1042$          RELEASE:       T3,1;
1090$          ASSIGN:        Treatment 3.NumberOut=Treatment 3.NumberOut + 1:
                              Treatment 3.WIP=Treatment 3.WIP-1:NEXT(83$);


;
;
;     Model statements for module:  BasicProcess.Assign 80 (Treatment Room3
Empty)
;
83$            ASSIGN:        TreatmentRoom3Status=0:NEXT(89$);


;
;
;     Model statements for module:  BasicProcess.Separate 25 (Separate after
treatment 3)
;
89$            SPLIT::NEXT(1093$);

1093$          ASSIGN:        Separate after treatment 3.NumberOut Orig=Separate
after treatment 3.NumberOut Orig + 1:NEXT(283$);
```

```
;
;
;     Model statements for module:  BasicProcess.Assign 142 (Assign location as
corridor 21)
;
283$          ASSIGN:        Track(rfid,2)=1:NEXT(209$);


;
;
;     Model statements for module:  BasicProcess.Decide 52 (Reroute entities
after treatment 3)
;
209$          BRANCH,        1:
                             If,Entity.Type==Equipment,210$,Yes:
                             If,Entity.Type==Patient,228$,Yes:
                             Else,209$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 56 (Station 56)
;

210$          STATION,       Station TR3ECR;
1100$         DELAY:         0.0,,VA:NEXT(211$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 39 (Route 39)
;
211$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station ECR;


;
;
;     Model statements for module:  AdvancedTransfer.Station 64 (Station 64)
;

228$          STATION,       Station TR3R;
1103$         DELAY:         0.0,,VA:NEXT(229$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 45 (Route 45)
;
229$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station R;


;
;
;     Model statements for module:  AdvancedTransfer.Station 30 (Station 30)
;

160$          STATION,       Station TR4;
1106$         DELAY:         0.0,,VA:NEXT(250$);


;
;
```

```
;     Model statements for module:  BasicProcess.Assign 113 (Assign location as
treatment room 4)
;
250$          ASSIGN:        Track(rfid,2)=11:NEXT(48$);


;
;
;     Model statements for module:  BasicProcess.Batch 24 (Wait for others
before treatment set up 4)
;
48$           QUEUE,         Wait for others before treatment set up 4.Queue;
1107$         GROUP,         ,Temporary:3,Last,Patient:NEXT(1108$);

1108$         ASSIGN:        Wait for others before treatment set up
4.NumberOut=
                             Wait for others before treatment set up
4.NumberOut + 1:NEXT(54$);


;
;
;     Model statements for module:  BasicProcess.Process 28 (Treatment Set Up
4)
;
54$           ASSIGN:        Treatment Set Up 4.NumberIn=Treatment Set Up
4.NumberIn + 1:
                             Treatment Set Up 4.WIP=Treatment Set Up 4.WIP+1;
1112$         QUEUE,         Treatment Set Up 4.Queue;
1111$         SEIZE,         2,VA:
                             TSU4,1:NEXT(1110$);

1110$         DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
1109$         RELEASE:       TSU4,1;
1157$         ASSIGN:        Treatment Set Up 4.NumberOut=Treatment Set Up
4.NumberOut + 1:
                             Treatment Set Up 4.WIP=Treatment Set Up 4.WIP-
1:NEXT(60$);


;
;
;     Model statements for module:  BasicProcess.Separate 20 (Separate after
treatment set up 4)
;
60$           SPLIT::NEXT(1160$);

1160$         ASSIGN:        Separate after treatment set up 4.NumberOut Orig=
                             Separate after treatment set up 4.NumberOut Orig +
1:NEXT(66$);


;
;
;     Model statements for module:  BasicProcess.Decide 44 (Reroute entities
after treatment set up 4)
;
66$           BRANCH,        1:
                             If,Entity.Type==Patient,72$,Yes:
                             If,Entity.Type==Equipment,72$,Yes:
                             Else,278$,Yes;

;
```

```
;
;     Model statements for module:  BasicProcess.Assign 137 (Assing location as
corridor 16)
;
278$          ASSIGN:        Track(rfid,2)=1:NEXT(198$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 51 (Station 51)
;

198$          STATION,       Station TR4NR;
1167$         DELAY:         0.0,,VA:NEXT(199$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 34 (Route 34)
;
199$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station NR;


;
;
;     Model statements for module:  BasicProcess.Batch 30 (Wait for other
before treatment 4)
;
72$           QUEUE,         Wait for other before treatment 4.Queue;
1168$         GROUP,         ,Temporary:2,Last,Patient:NEXT(1169$);

1169$         ASSIGN:        Wait for other before treatment 4.NumberOut=Wait
for other before treatment 4.NumberOut + 1
                             :NEXT(78$);


;
;
;     Model statements for module:  BasicProcess.Process 34 (Treatment 4)
;
78$           ASSIGN:        Treatment 4.NumberIn=Treatment 4.NumberIn + 1:
                             Treatment 4.WIP=Treatment 4.WIP+1;
1173$         QUEUE,         Treatment 4.Queue;
1172$         SEIZE,         2,VA:
                             T4,1:NEXT(1171$);

1171$         DELAY:         MinutesToBaseTime(Normal(30,3)),,VA;
1170$         RELEASE:       T4,1;
1218$         ASSIGN:        Treatment 4.NumberOut=Treatment 4.NumberOut + 1:
                             Treatment 4.WIP=Treatment 4.WIP-1:NEXT(84$);


;
;
;     Model statements for module:  BasicProcess.Assign 81 (Treatment Room4
Empty)
;
84$           ASSIGN:        TreatmentRoom4Status=0:NEXT(90$);


;
;
```

```
;     Model statements for module:  BasicProcess.Separate 26 (Separate after
treatment 4)
;
90$           SPLIT::NEXT(1221$);

1221$         ASSIGN:        Separate after treatment 4.NumberOut Orig=Separate
after treatment 4.NumberOut Orig + 1:NEXT(284$);


;
;
;     Model statements for module:  BasicProcess.Assign 143 (Assign location as
corridor 22)
;
284$          ASSIGN:        Track(rfid,2)=1:NEXT(212$);


;
;
;     Model statements for module:  BasicProcess.Decide 53 (Reroute entities
after treatment 4)
;
212$          BRANCH,        1:
                             If,Entity.Type==Equipment,213$,Yes:
                             If,Entity.Type==Patient,230$,Yes:
                             Else,212$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 57 (Station 57)
;

213$          STATION,       Station TR4ECR;
1228$         DELAY:         0.0,,VA:NEXT(214$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 40 (Route 40)
;
214$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station ECR;


;
;
;     Model statements for module:  AdvancedTransfer.Station 65 (Station 65)
;

230$          STATION,       Station TR4R;
1231$         DELAY:         0.0,,VA:NEXT(231$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 46 (Route 46)
;
231$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station R;


;
;
;     Model statements for module:  AdvancedTransfer.Station 32 (Station 32)
;
```

```
163$          STATION,       Station TR5;
1234$         DELAY:         0.0,,VA:NEXT(251$);


;
;
;     Model statements for module:  BasicProcess.Assign 114 (Assign location as
treatment room 5)
;
251$          ASSIGN:        Track(rfid,2)=12:NEXT(49$);


;
;
;     Model statements for module:  BasicProcess.Batch 25 (Wait for others
before treatment set up 5)
;
49$           QUEUE,         Wait for others before treatment set up 5.Queue;
1235$         GROUP,         ,Temporary:3,Last,Patient:NEXT(1236$);

1236$         ASSIGN:        Wait for others before treatment set up
5.NumberOut=
                             Wait for others before treatment set up
5.NumberOut + 1:NEXT(55$);


;
;
;     Model statements for module:  BasicProcess.Process 29 (Treatment Set Up
5)
;
55$           ASSIGN:        Treatment Set Up 5.NumberIn=Treatment Set Up
5.NumberIn + 1:
                             Treatment Set Up 5.WIP=Treatment Set Up 5.WIP+1;
1240$         QUEUE,         Treatment Set Up 5.Queue;
1239$         SEIZE,         2,VA:
                             TSU5,1:NEXT(1238$);

1238$         DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
1237$         RELEASE:       TSU5,1;
1285$         ASSIGN:        Treatment Set Up 5.NumberOut=Treatment Set Up
5.NumberOut + 1:
                             Treatment Set Up 5.WIP=Treatment Set Up 5.WIP-
1:NEXT(61$);


;
;
;     Model statements for module:  BasicProcess.Separate 21 (Separate after
treatment set up 5)
;
61$           SPLIT::NEXT(1288$);

1288$         ASSIGN:        Separate after treatment set up 5.NumberOut Orig=
                             Separate after treatment set up 5.NumberOut Orig +
1:NEXT(67$);


;
;
;     Model statements for module:  BasicProcess.Decide 45 (Reroute entities
after treatment set up 5)
```

```
;
67$           BRANCH,        1:
                             If,Entity.Type==Patient,73$,Yes:
                             If,Entity.Type==Equipment,73$,Yes:
                             Else,279$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 138 (Assing location as
corridor 17)
;
279$          ASSIGN:        Track(rfid,2)=1:NEXT(200$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 52 (Station 52)
;

200$          STATION,       Station TR5NR;
1295$         DELAY:         0.0,,VA:NEXT(201$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 35 (Route 35)
;
201$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station NR;


;
;
;     Model statements for module:  BasicProcess.Batch 31 (Wait for other
before treatment 5)
;
73$           QUEUE,         Wait for other before treatment 5.Queue;
1296$         GROUP,         ,Temporary:2,Last,Patient:NEXT(1297$);

1297$         ASSIGN:        Wait for other before treatment 5.NumberOut=Wait
for other before treatment 5.NumberOut + 1
                             :NEXT(79$);


;
;
;     Model statements for module:  BasicProcess.Process 35 (Treatment 5)
;
79$           ASSIGN:        Treatment 5.NumberIn=Treatment 5.NumberIn + 1:
                             Treatment 5.WIP=Treatment 5.WIP+1;
1301$         QUEUE,         Treatment 5.Queue;
1300$         SEIZE,         2,VA:
                             T5,1:NEXT(1299$);

1299$         DELAY:         MinutesToBaseTime(Normal(30,3)),,VA;
1298$         RELEASE:       T5,1;
1346$         ASSIGN:        Treatment 5.NumberOut=Treatment 5.NumberOut + 1:
                             Treatment 5.WIP=Treatment 5.WIP-1:NEXT(85$);


;
;
;     Model statements for module:  BasicProcess.Assign 82 (Treatment Room5
Empty)
```

```
;
85$           ASSIGN:        TreatmentRoom5Status=0:NEXT(91$);


;
;
;     Model statements for module:  BasicProcess.Separate 27 (Separate after
treatment 5)
;
91$           SPLIT::NEXT(1349$);

1349$         ASSIGN:        Separate after treatment 5.NumberOut Orig=Separate
after treatment 5.NumberOut Orig + 1:NEXT(285$);


;
;
;     Model statements for module:  BasicProcess.Assign 144 (Assign location as
corridor 23)
;
285$          ASSIGN:        Track(rfid,2)=1:NEXT(215$);


;
;
;     Model statements for module:  BasicProcess.Decide 54 (Reroute entities
after treatment 5)
;
215$          BRANCH,        1:
                             If,Entity.Type==Equipment,216$,Yes:
                             If,Entity.Type==Patient,232$,Yes:
                             Else,215$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 58 (Station 58)
;

216$          STATION,       Station TR5ECR;
1356$         DELAY:         0.0,,VA:NEXT(217$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 41 (Route 41)
;
217$          ROUTE:         MinutesToBaseTime(NORM( 3, 0.3 )),Station ECR;


;
;
;     Model statements for module:  AdvancedTransfer.Station 66 (Station 66)
;

232$          STATION,       Station TR5R;
1359$         DELAY:         0.0,,VA:NEXT(233$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 47 (Route 47)
;
233$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station R;
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Station 34 (Station 34)
;

166$          STATION,       Station TR6;
1362$         DELAY:         0.0,,VA:NEXT(252$);


;
;
;     Model statements for module:  BasicProcess.Assign 115 (Assign location as
treatment room 6)
;
252$          ASSIGN:        Track(rfid,2)=13:NEXT(50$);


;
;
;     Model statements for module:  BasicProcess.Batch 26 (Wait for others
before treatment set up 6)
;
50$           QUEUE,         Wait for others before treatment set up 6.Queue;
1363$         GROUP,         ,Temporary:3,Last,Patient:NEXT(1364$);

1364$         ASSIGN:        Wait for others before treatment set up
6.NumberOut=
                             Wait for others before treatment set up
6.NumberOut + 1:NEXT(56$);


;
;
;     Model statements for module:  BasicProcess.Process 30 (Treatment Set Up
6)
;
56$           ASSIGN:        Treatment Set Up 6.NumberIn=Treatment Set Up
6.NumberIn + 1:
                             Treatment Set Up 6.WIP=Treatment Set Up 6.WIP+1;
1368$         QUEUE,         Treatment Set Up 6.Queue;
1367$         SEIZE,         2,VA:
                             TSU6,1:NEXT(1366$);

1366$         DELAY:         MinutesToBaseTime(Normal(10,1)),,VA;
1365$         RELEASE:       TSU6,1;
1413$         ASSIGN:        Treatment Set Up 6.NumberOut=Treatment Set Up
6.NumberOut + 1:
                             Treatment Set Up 6.WIP=Treatment Set Up 6.WIP-
1:NEXT(62$);


;
;
;     Model statements for module:  BasicProcess.Separate 22 (Separate after
treatment set up 6)
;
62$           SPLIT::NEXT(1416$);

1416$         ASSIGN:        Separate after treatment set up 6.NumberOut Orig=
                             Separate after treatment set up 6.NumberOut Orig +
1:NEXT(68$);
```

```
;
;
;     Model statements for module:  BasicProcess.Decide 46 (Reroute entities
after treatment set up 6)
;
68$           BRANCH,        1:
                             If,Entity.Type==Patient,74$,Yes:
                             If,Entity.Type==Equipment,74$,Yes:
                             Else,280$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 139 (Assing location as
corridor 18)
;
280$          ASSIGN:        Track(rfid,2)=1:NEXT(202$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 53 (Station 53)
;

202$          STATION,       Station TR6NR;
1423$         DELAY:         0.0,,VA:NEXT(203$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 36 (Route 36)
;
203$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station NR;


;
;
;     Model statements for module:  BasicProcess.Batch 32 (Wait for other
before treatment 6)
;
74$           QUEUE,         Wait for other before treatment 6.Queue;
1424$         GROUP,         ,Temporary:2,Last,Patient:NEXT(1425$);

1425$         ASSIGN:        Wait for other before treatment 6.NumberOut=Wait
for other before treatment 6.NumberOut + 1
                             :NEXT(80$);


;
;
;     Model statements for module:  BasicProcess.Process 36 (Treatment 6)
;
80$           ASSIGN:        Treatment 6.NumberIn=Treatment 6.NumberIn + 1:
                             Treatment 6.WIP=Treatment 6.WIP+1;
1429$         QUEUE,         Treatment 6.Queue;
1428$         SEIZE,         2,VA:
                             T6,1:NEXT(1427$);

1427$         DELAY:         MinutesToBaseTime(Normal(30,3)),,VA;
1426$         RELEASE:       T6,1;
1474$         ASSIGN:        Treatment 6.NumberOut=Treatment 6.NumberOut + 1:
                             Treatment 6.WIP=Treatment 6.WIP-1:NEXT(86$);
```

```
;
;
;     Model statements for module:  BasicProcess.Assign 83 (Treatment Room6
Empty)
;
86$           ASSIGN:        TreatmentRoom6Status=0:NEXT(92$);


;
;
;     Model statements for module:  BasicProcess.Separate 28 (Separate after
treatment 6)
;
92$           SPLIT::NEXT(1477$);

1477$         ASSIGN:        Separate after treatment 6.NumberOut Orig=Separate
after treatment 6.NumberOut Orig + 1:NEXT(286$);


;
;
;     Model statements for module:  BasicProcess.Assign 145 (Assign location as
corridor 24)
;
286$          ASSIGN:        Track(rfid,2)=1:NEXT(218$);


;
;
;     Model statements for module:  BasicProcess.Decide 55 (Reroute entities
after treatment 6)
;
218$          BRANCH,        1:
                             If,Entity.Type==Equipment,219$,Yes:
                             If,Entity.Type==Patient,234$,Yes:
                             Else,218$,Yes;

;
;
;     Model statements for module:  AdvancedTransfer.Station 59 (Station 59)
;

219$          STATION,       Station TR6ECR;
1484$         DELAY:         0.0,,VA:NEXT(220$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 42 (Route 42)
;
220$          ROUTE:         MinutesToBaseTime(NORM( 3 , 0.3 )),Station ECR;


;
;
;     Model statements for module:  AdvancedTransfer.Station 67 (Station 67)
;

234$          STATION,       Station TR6R;
1487$         DELAY:         0.0,,VA:NEXT(235$);
```

```
;
;
;     Model statements for module:  AdvancedTransfer.Route 48 (Route 48)
;
235$          ROUTE:         MinutesToBaseTime(NORM( 2 , 0.2 )),Station R;


;
;
;     Model statements for module:  AdvancedTransfer.Station 48 (Station 48)
;

193$          STATION,       Station NR;
1490$         DELAY:         0.0,,VA:NEXT(240$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 60 (Station 60)
;

221$          STATION,       Station ECR;
1493$         DELAY:         0.0,,VA:NEXT(254$);


;
;
;     Model statements for module:  BasicProcess.Assign 117 (Assign location as
equipment cleaning room)
;
254$          ASSIGN:        Track(rfid,2)=15:NEXT(114$);


;
;
;     Model statements for module:  BasicProcess.Assign 99 (Calculate cleaning
time)
;
114$          ASSIGN:        cleaningmeantime=120+120*EP(-
0.06*MX(equipmentcleaned,0)):
                             EquipmentCleaned=EquipmentCleaned+1:NEXT(113$);


;
;
;     Model statements for module:  BasicProcess.Process 38 (Equipment
cleaning)
;
113$          ASSIGN:        Equipment cleaning.NumberIn=Equipment
cleaning.NumberIn + 1:
                             Equipment cleaning.WIP=Equipment cleaning.WIP+1;
1497$         QUEUE,         Equipment cleaning.Queue;
1496$         SEIZE,         2,VA:
                             ECR,1:NEXT(1495$);

1495$         DELAY:
MinutesToBaseTime(NORM(cleaningmeantime,cleaningmeantime*0.1,1)),,VA;
1494$         RELEASE:       ECR,1;
1542$         ASSIGN:        Equipment cleaning.NumberOut=Equipment
cleaning.NumberOut + 1:
                             Equipment cleaning.WIP=Equipment cleaning.WIP-
1:NEXT(288$);
```

```
;
;
;     Model statements for module:  BasicProcess.Assign 147 (Assign location as
corridor 26)
;
288$          ASSIGN:        Track(rfid,2)=1:NEXT(236$);


;
;
;     Model statements for module:  AdvancedTransfer.Station 68 (Station 68)
;

236$          STATION,       Station ECRER1;
1547$         DELAY:         0.0,,VA:NEXT(237$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 49 (Route 49)
;
237$          ROUTE:         MinutesToBaseTime(NORM( 5 , 0.5 )),Station ECRER2;


;
;
;     Model statements for module:  AdvancedTransfer.Station 62 (Station 62)
;

224$          STATION,       Station R;
1550$         DELAY:         0.0,,VA:NEXT(253$);


;
;
;     Model statements for module:  BasicProcess.Assign 116 (Assign location as
register 3)
;
253$          ASSIGN:        Track(rfid,2)=2:NEXT(225$);


;
;
;     Model statements for module:  BasicProcess.Process 41 (Register 4)
;
225$          ASSIGN:        Register 4.NumberIn=Register 4.NumberIn + 1:
                             Register 4.WIP=Register 4.WIP+1;
1554$         QUEUE,         Register 4.Queue;
1553$         SEIZE,         2,VA:
                             registrar,1:NEXT(1552$);

1552$         DELAY:         MinutesToBaseTime(Normal(1,0.1)),,VA;
1551$         RELEASE:       registrar,1;
1599$         ASSIGN:        Register 4.NumberOut=Register 4.NumberOut + 1:
                             Register 4.WIP=Register 4.WIP-1:NEXT(112$);


;
;
;     Model statements for module:  BasicProcess.Assign 98 (Release Tag 2)
;
```

```
112$          ASSIGN:        Track(rfid,1)=0:NEXT(111$);


;
;
;     Model statements for module:  BasicProcess.Dispose 11 (Departure after
treatment)
;
111$          ASSIGN:        Departure after treatment.NumberOut=Departure
after treatment.NumberOut + 1;
1602$         DISPOSE:       Yes;


;
;
;     Model statements for module:  AdvancedTransfer.Station 69 (Station 69)
;

238$          STATION,       Station ECRER2;
1605$         DELAY:         0.0,,VA:NEXT(241$);


;
;
;     Model statements for module:  BasicProcess.Create 10 (Create Tracking
Token)
;

1606$         CREATE,
1,HoursToBaseTime(0.001),Tracker:HoursToBaseTime(EXPO(1)),1:NEXT(1607$);

1607$         ASSIGN:        Create Tracking Token.NumberOut=Create Tracking
Token.NumberOut + 1:NEXT(258$);


;
;
;     Model statements for module:  BasicProcess.Decide 57 (All tags reported?)
;
258$          BRANCH,        1:
                             If,TrackerCounter==129,1610$,Yes:
                             Else,1611$,Yes;
1610$         ASSIGN:        All tags reported?.NumberOut True=All tags
reported?.NumberOut True + 1:NEXT(259$);

1611$         ASSIGN:        All tags reported?.NumberOut False=All tags
reported?.NumberOut False + 1:NEXT(256$);


;
;
;     Model statements for module:  BasicProcess.Assign 119 (Reset tracker
counter)
;
259$          ASSIGN:        TrackerCounter=1:NEXT(260$);


;
;
;     Model statements for module:  BasicProcess.Process 42 (Wait 10 seconds)
;
260$          ASSIGN:        Wait 10 seconds.NumberIn=Wait 10 seconds.NumberIn
+ 1:
```

```
                             Wait 10 seconds.WIP=Wait 10 seconds.WIP+1;
1613$         DELAY:         10,,VA;
1660$         ASSIGN:        Wait 10 seconds.NumberOut=Wait 10
seconds.NumberOut + 1:
                             Wait 10 seconds.WIP=Wait 10 seconds.WIP-
1:NEXT(258$);


;
;
;     Model statements for module:  BasicProcess.Decide 56 (Is the tag active?)
;
256$          BRANCH,        1:
                             If,Track(TrackerCounter,1)==1,1663$,Yes:
                             Else,1664$,Yes;
1663$         ASSIGN:        Is the tag active?.NumberOut True=Is the tag
active?.NumberOut True + 1:NEXT(297$);

1664$         ASSIGN:        Is the tag active?.NumberOut False=Is the tag
active?.NumberOut False + 1:NEXT(257$);


;
;
;     Model statements for module:  BasicProcess.Decide 65 (Add tracking error
under these conditions)
;
297$          BRANCH,        1:
                             If,TrackerCounter>110 &&
TrackerCounter<113&&CalHour(tnow)>6,1665$,Yes:
                             Else,1666$,Yes;
1665$         ASSIGN:        Add tracking error under these
conditions.NumberOut True=
                             Add tracking error under these
conditions.NumberOut True + 1:NEXT(290$);

1666$         ASSIGN:        Add tracking error under these
conditions.NumberOut False=
                             Add tracking error under these
conditions.NumberOut False + 1:NEXT(262$);


;
;
;     Model statements for module:  BasicProcess.Decide 60 (Add errors?)
;
290$          BRANCH,        1:
                             If,Errors==1,1667$,Yes:
                             Else,1668$,Yes;
1667$         ASSIGN:        Add errors?.NumberOut True=Add errors?.NumberOut
True + 1:NEXT(289$);

1668$         ASSIGN:        Add errors?.NumberOut False=Add errors?.NumberOut
False + 1:NEXT(262$);


;
;
;     Model statements for module:  BasicProcess.Decide 59 (Check error
probability)
;
289$          BRANCH,        1:
                             With,(98)/100,262$,Yes:
```

```
                             With,(1)/100,293$,Yes:
                             With,(1)/100,296$,Yes:
                             Else,289$,Yes;

;
;
;     Model statements for module:  BasicProcess.Decide 58 (Did the tag change
its location? 1)
;
262$          BRANCH,        1:
                             If,Places( Track( TrackerCounter, 2)
,1)==PreviousLocation(TrackerCounter)&&CompactData,1671$,Yes:
                             Else,1672$,Yes;
1671$         ASSIGN:        Did the tag change its location? 1.NumberOut True=
                             Did the tag change its location? 1.NumberOut True
+ 1:NEXT(257$);

1672$         ASSIGN:        Did the tag change its location? 1.NumberOut
False=
                             Did the tag change its location? 1.NumberOut False
+ 1:NEXT(255$);


;
;
;     Model statements for module:  BasicProcess.Assign 118 (Iterate through
tags)
;
257$          ASSIGN:        TrackerCounter=TrackerCounter+1:NEXT(258$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 39 (Report
location)
;
255$          WRITE,         File 1,RECORDSET(Recordset 1):
                             TrackerCounter,
                             Places( Track( TrackerCounter, 2) ,1),


str(CalMonth(TNOW))+"/"+str(CalDayOfMonth(TNOW))+"/"+str(CalYear(TNOW))+"
"+str(CalHour(TNOW))+":"+str(CalMinute(TNOW))+":"+str(CalSecond(TNOW))
                             :NEXT(261$);


;
;
;     Model statements for module:  BasicProcess.Assign 120 (Record location)
;
261$          ASSIGN:        PreviousLocation(TrackerCounter)=Places( Track(
TrackerCounter, 2) ,1):NEXT(257$);


;
;
;     Model statements for module:  BasicProcess.Decide 63 (Did the tag change
its location? 2)
;
293$          BRANCH,        1:
                             If,Places( Track( TrackerCounter, 2)
,2)==PreviousLocation(TrackerCounter)&&CompactData,1673$,Yes:
                             Else,1674$,Yes;
```

```
1673$         ASSIGN:        Did the tag change its location? 2.NumberOut True=
                             Did the tag change its location? 2.NumberOut True
+ 1:NEXT(257$);

1674$         ASSIGN:        Did the tag change its location? 2.NumberOut
False=
                             Did the tag change its location? 2.NumberOut False
+ 1:NEXT(291$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 42 (Report
location 2)
;
291$          WRITE,         File 1,RECORDSET(Recordset 1):
                             TrackerCounter,
                             Places( Track( TrackerCounter, 2) ,2),


str(CalMonth(TNOW))+"/"+str(CalDayOfMonth(TNOW))+"/"+str(CalYear(TNOW))+"
"+str(CalHour(TNOW))+":"+str(CalMinute(TNOW))+":"+str(CalSecond(TNOW))
                             :NEXT(292$);


;
;
;     Model statements for module:  BasicProcess.Assign 150 (Record location 2)
;
292$          ASSIGN:        PreviousLocation(TrackerCounter)=Places( Track(
TrackerCounter, 2) ,2):NEXT(257$);


;
;
;     Model statements for module:  BasicProcess.Decide 64 (Did the tag change
its location? 3)
;
296$          BRANCH,        1:
                             If,Places( Track( TrackerCounter, 2)
,3)==PreviousLocation(TrackerCounter)&&CompactData,1675$,Yes:
                             Else,1676$,Yes;
1675$         ASSIGN:        Did the tag change its location? 3.NumberOut True=
                             Did the tag change its location? 3.NumberOut True
+ 1:NEXT(257$);

1676$         ASSIGN:        Did the tag change its location? 3.NumberOut
False=
                             Did the tag change its location? 3.NumberOut False
+ 1:NEXT(294$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 43 (Report
location 3)
;
294$          WRITE,         File 1,RECORDSET(Recordset 1):
                             TrackerCounter,
                             Places( Track( TrackerCounter, 2) ,3),
str(CalMonth(TNOW))+"/"+str(CalDayOfMonth(TNOW))+"/"+str(CalYear(TNOW))+"
"+str(CalHour(TNOW))+":"+str(CalMinute(TNOW))+":"+str(CalSecond(TNOW))
                              :NEXT(295$);


;
;
;     Model statements for module:  BasicProcess.Assign 151 (Record location 3)
;
295$          ASSIGN:        PreviousLocation(TrackerCounter)=Places( Track(
TrackerCounter, 2) ,3):NEXT(257$);
```

```
PROJECT,       "Unnamed Project","Cenk",,,No,Yes,Yes,Yes,No,No,No,No,No,No;

ATTRIBUTES:    RFID:
               PatientID;

FILES:         File 1,
               "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\Cenk\Desktop\Healthcare\simulation model\Hospital.accdb;Persist
Security Info=False;",
               ADO,,Dispose,,Hold,RECORDSET(Recordset 1,"select RFID_NO,
Location_ID, Time_Stamp from tracking",-1),RECORDSET(Recordset 2,"select
Patient_ID, RFID from visit",-1);

VARIABLES:     Is the tag active?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
               Treatment 2.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
               Treatment 4.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
               TreatmentRoom5NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
               Treatment 6.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
               Separate after treatment 4.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
               Separate after consultation 1.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
               Treatment Set Up
1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Separate after treatment set up 5.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
               Choose Tag.NumberOut True,CLEAR(Statistics),CATEGORY("Exclude"):
               EquipmentCleaned,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real),-50:
               Wait 10 seconds.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Create Equipment.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               TreatmentRoom1Status,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
               TreatmentRoom6Status,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
               Treatment Set Up 6.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
               TreatmentRoom1NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
               Wait for other before treatment
3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Add tracking error under these conditions.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
               Treatment 2.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
               TrackerCounter,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real),1:
               TreatmentRoom1EquipmentRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
               Register 4.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
               Treatment Set Up 1.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
               PreviousLocation(128),CLEAR(System),DATATYPE(String):
               Did the tag change its location? 2.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
               Wait for other before treatment
6.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               ConsultationRoom1NeedDoc,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
```

```
              TreatmentRoom5EquipmentRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              TreatmentRoom6NeedEquipment,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Wait for others before treatment set up
5.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment 1.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              Register 2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Create Nurse.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Create Doctor.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment set up 2.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              ConsultationRoom2DoctorRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Treatment Set Up
6.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait 10 seconds.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Treatment 4.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              cleaningmeantime,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Wait for others before treatment set up
2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom1NeedNurse,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              TreatmentRoom3NeedNurse,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              TreatmentRoom6NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              TreatmentRoom5NeedNurse,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Separate after treatment 5.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after consultation 2.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              Consultation 2.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Did the tag change its location? 3.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 4.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Treatment Set Up 3.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Did the tag change its location? 1.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom2Status,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Treatment Set Up
3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Register 1.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              TreatmentRoom5NeedEquipment,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Treatment 1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom2EquipmentRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              TreatmentRoom2NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Add tracking error under these conditions.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Register 2.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up
1.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
```

```
              Register 4.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom6EquipmentRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Consultation 1.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Treatment Set Up
3.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom4NeedEquipment,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Treatment Set Up
5.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for other before treatment
5.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment set up 6.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              Equipment
cleaning.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              ConsultationRoom1Status,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Separate after treatment 2.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              ConsultationRoom1NeedNurse,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              ConsultationRoom1NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Register 1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment set up 3.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom3Status,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Treatment 6.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Add errors?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 1.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Register 4.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 3.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Choose Tag.NumberOut False,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up 5.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Equipment
cleaning.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 5.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for other before treatment
2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 1.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              ConsultationRoom1DoctorRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Register 3.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Errors,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Wait 10 seconds.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom3NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Treatment 6.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for others before treatment set up
1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment 6.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom3NeedEquipment,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
```

```
              Wait for others before treatment set up
4.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom3EquipmentRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              ConsultationRoom2NeedDoc,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Treatment Set Up
2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for others before consultation
2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up
5.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Consultation 2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom4Status,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              ConsultationRoom2NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):

Places(15,3),CLEAR(System),DATATYPE(String),"Corridor","Register","Waiting
Room","Consultation Room 1",
              "Consultation Room 2","Nurses Room","Doctors Room","Treatment
Room 1","Treatment Room 2","Treatment Room 3",
              "Treatment Room 4","Treatment Room 5","Treatment Room
6","Equipment Room","Equipment Cleaning","Corridor",
              "Register","Consultation Room 1","Consultation Room
2","Consultation Room 1","Doctors Room","Nurses Room",
              "Treatment Room 2","Treatment Room 3","Treatment Room
4","Treatment Room 5","Treatment Room 6","Treatment Room 1",
              "Equipment Cleaning","Equipment
Room","Corridor","Register","Corridor":
              RFIDCounter,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real),1:
              Did the tag change its location? 2.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 3.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Treatment Set Up 2.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Go to treatment or leave?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              ConsultationRoom2Status,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Go to treatment or leave?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment 3.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom2NeedEquipment,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Separate after treatment set up 4.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              Departure after
consultation.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom2NeedNurse,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Is the tag active?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom4NeedNurse,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Equipment cleaning.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Consultation 1.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
```

```
              TreatmentRoom6NeedNurse,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Wait for other before treatment
1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for other before treatment
4.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom4NurseRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              EquipmentCounter,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Did the tag change its location? 1.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for others before treatment set up
3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Wait for others before treatment set up
6.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Register 1.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              CompactData,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real),1:
              Add errors?.NumberOut True,CLEAR(Statistics),CATEGORY("Exclude"):
              Register 3.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up
2.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              NurseCounter,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Treatment 5.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Register 3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Create Patient.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up
4.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up 4.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Track(128,2),CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Treatment Set Up
6.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom1NeedEquipment,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              All tags reported?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom5Status,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Register 2.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Wait for others before consultation
1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment Set Up
4.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              TreatmentRoom4EquipmentRequestTime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Consultation 1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Separate after treatment set up 1.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              All tags reported?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Treatment 5.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              DoctorCounter,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              ConsultationRoom2NeedNurse,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
```

```
              Did the tag change its location? 3.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Create Tracking
Token.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              PatientCounter,CLEAR(System),CATEGORY("User Specified-User
Specified"),DATATYPE(Real):
              Departure after
treatment.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Consultation 2.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real);

SEEDS:        1,21121955,Yes;

QUEUES:       Wait for other before treatment 4.Queue,FIFO,,AUTOSTATS(Yes,,):
              Consultation 2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Doctors wait for a call.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before consultation
1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment Set Up 5.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before treatment set up
6.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment 5.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before treatment set up
1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment Set Up 1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Register 3.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment Set Up 6.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for other before treatment 5.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment 6.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for available treatment room.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before consultation
2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before treatment set up
2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment 1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Register 4.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for other before treatment 1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for other before treatment 6.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment Set Up 2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before treatment set up
3.Queue,FIFO,,AUTOSTATS(Yes,,):
              Equipment cleaning.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment 2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for available consultation
room.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment Set Up 3.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for other before treatment 2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Equipment wait for a call.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment 3.Queue,FIFO,,AUTOSTATS(Yes,,):
              Consultation 1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Register 1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before treatment set up
4.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment Set Up 4.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for other before treatment 3.Queue,FIFO,,AUTOSTATS(Yes,,):
              Treatment 4.Queue,FIFO,,AUTOSTATS(Yes,,):
              Nurses wait for a call.Queue,FIFO,,AUTOSTATS(Yes,,):
              Wait for others before treatment set up
5.Queue,FIFO,,AUTOSTATS(Yes,,);

PICTURES:     Picture.Airplane:
              Picture.Green Ball:
              Picture.Blue Page:
```

```
              Picture.Telephone:
              Picture.Blue Ball:
              Picture.Yellow Page:
              Picture.EMail:
              People.Man.Healthcare:
              Picture.Yellow Ball:
              Picture.Bike:
              Picture.Report:
              Picture.Van:
              Picture.Widgets:
              Picture.Envelope:
              Picture.Fax:
              Picture.Truck:
              Picture.Person:
              Picture.Letter:
              People.Woman.Healthcare:
              Picture.Box:
              Equipment.Crutches:
              Picture.Woman:
              Picture.Package:
              Picture.Man:
              Picture.Diskette:
              Picture.Boat:
              People.Patient:
              Picture.Red Page:
              Picture.Ball:
              Picture.Green Page:
              Picture.Red Ball;

RESOURCES:
registrar,Capacity(2),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,)
:

ECR,Capacity(Infinite),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,
):

TSU1,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

TSU2,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

TSU3,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

TSU4,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

TSU5,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

TSU6,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

T1,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

T2,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

T3,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

T4,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

T5,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

T6,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

CR1,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,):

CR2,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTOSTATS(Yes,,);
```

```
STATIONS:     Station ECR,,,Station ECR,AUTOSTATS(Yes,,):
              Station CRR1,,,Station CRR1,AUTOSTATS(Yes,,):
              Station CRR2,,,Station CRR2,AUTOSTATS(Yes,,):
              Station NRCR11,,,Station NRCR11,AUTOSTATS(Yes,,):
              Station NRCR12,,,Station NRCR12,AUTOSTATS(Yes,,):
              Station TR2NR,,,Station TR2NR,AUTOSTATS(Yes,,):
              Station NRCR21,,,Station NRCR21,AUTOSTATS(Yes,,):
              Station NRCR22,,,Station NRCR22,AUTOSTATS(Yes,,):
              Station RWR1,,,Station RWR1,AUTOSTATS(Yes,,):
              Station RWR2,,,Station RWR2,AUTOSTATS(Yes,,):
              Station TR1R,,,Station TR1R,AUTOSTATS(Yes,,):
              Station TR2R,,,Station TR2R,AUTOSTATS(Yes,,):
              Station TR3NR,,,Station TR3NR,AUTOSTATS(Yes,,):
              Station WRTR1,,,Station WRTR1,AUTOSTATS(Yes,,):
              Station WRTR2,,,Station WRTR2,AUTOSTATS(Yes,,):
              Station CRNR1,,,Station CRNR1,AUTOSTATS(Yes,,):
              Station WRTR3,,,Station WRTR3,AUTOSTATS(Yes,,):
              Station CRNR2,,,Station CRNR2,AUTOSTATS(Yes,,):
              Station WRTR4,,,Station WRTR4,AUTOSTATS(Yes,,):
              Station WRTR5,,,Station WRTR5,AUTOSTATS(Yes,,):
              Station WRTR6,,,Station WRTR6,AUTOSTATS(Yes,,):
              Station TR3R,,,Station TR3R,AUTOSTATS(Yes,,):
              Station TR4R,,,Station TR4R,AUTOSTATS(Yes,,):
              Station ECRER1,,,Station ECRER1,AUTOSTATS(Yes,,):
              Station ECRER2,,,Station ECRER2,AUTOSTATS(Yes,,):
              Station TR1ECR,,,Station TR1ECR,AUTOSTATS(Yes,,):
              Station R,,,Station R,AUTOSTATS(Yes,,):
              Station TR2ECR,,,Station TR2ECR,AUTOSTATS(Yes,,):
              Station CRDR1,,,Station CRDR1,AUTOSTATS(Yes,,):
              Station TR4NR,,,Station TR4NR,AUTOSTATS(Yes,,):
              Station CRDR2,,,Station CRDR2,AUTOSTATS(Yes,,):
              Station TR5R,,,Station TR5R,AUTOSTATS(Yes,,):
              Station TR3ECR,,,Station TR3ECR,AUTOSTATS(Yes,,):
              Station TR4ECR,,,Station TR4ECR,AUTOSTATS(Yes,,):
              Station TR5ECR,,,Station TR5ECR,AUTOSTATS(Yes,,):
              Station NR,,,Station NR,AUTOSTATS(Yes,,):
              Station TR6ECR,,,Station TR6ECR,AUTOSTATS(Yes,,):
              Station TR6R,,,Station TR6R,AUTOSTATS(Yes,,):
              Station ERTR1,,,Station ERTR1,AUTOSTATS(Yes,,):
              Station ERTR2,,,Station ERTR2,AUTOSTATS(Yes,,):
              Station ERTR3,,,Station ERTR3,AUTOSTATS(Yes,,):
              Station ERTR4,,,Station ERTR4,AUTOSTATS(Yes,,):
              Station ERTR5,,,Station ERTR5,AUTOSTATS(Yes,,):
              Station ERTR6,,,Station ERTR6,AUTOSTATS(Yes,,):
              Station DRCR11,,,Station DRCR11,AUTOSTATS(Yes,,):
              Station DRCR12,,,Station DRCR12,AUTOSTATS(Yes,,):
              Station TR1,,,Station TR1,AUTOSTATS(Yes,,):
              Station TR5NR,,,Station TR5NR,AUTOSTATS(Yes,,):
              Station TR2,,,Station TR2,AUTOSTATS(Yes,,):
              Station TR3,,,Station TR3,AUTOSTATS(Yes,,):
              Station TR4,,,Station TR4,AUTOSTATS(Yes,,):
              Station TR5,,,Station TR5,AUTOSTATS(Yes,,):
              Station TR6,,,Station TR6,AUTOSTATS(Yes,,):
              Station DRCR21,,,Station DRCR21,AUTOSTATS(Yes,,):
              Station DRCR22,,,Station DRCR22,AUTOSTATS(Yes,,):
              Station WRCR11,,,Station WRCR11,AUTOSTATS(Yes,,):
              Station WRCR12,,,Station WRCR12,AUTOSTATS(Yes,,):
              Station TR6NR,,,Station TR6NR,AUTOSTATS(Yes,,):
              Station WRCR21,,,Station WRCR21,AUTOSTATS(Yes,,):
              Station WRCR22,,,Station WRCR22,AUTOSTATS(Yes,,):
              Station TR1NR,,,Station TR1NR,AUTOSTATS(Yes,,):
              Station NRTR1,,,Station NRTR1,AUTOSTATS(Yes,,):
```

```
              Station CRWR1,,,Station CRWR1,AUTOSTATS(Yes,,):
              Station NRTR2,,,Station NRTR2,AUTOSTATS(Yes,,):
              Station CRWR2,,,Station CRWR2,AUTOSTATS(Yes,,):
              Station NRTR3,,,Station NRTR3,AUTOSTATS(Yes,,):
              Station NRTR4,,,Station NRTR4,AUTOSTATS(Yes,,):
              Station NRTR5,,,Station NRTR5,AUTOSTATS(Yes,,):
              Station NRTR6,,,Station NRTR6,AUTOSTATS(Yes,,);

REPLICATE,    1,,HoursToBaseTime(30),Yes,Yes,,,,24,Seconds,No,No,,,Yes;

ENTITIES:     Tracker,Picture.Report,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,):

Equipment,Equipment.Crutches,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,):
              Patient,People.Patient,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,):

Nurse,People.Woman.Healthcare,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,):

Doctor,People.Man.Healthcare,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,);

ACTIVITYAREAS: Station ECR,0,,AUTOSTATS(Yes,,):
              Station CRR1,0,,AUTOSTATS(Yes,,):
              Station CRR2,0,,AUTOSTATS(Yes,,):
              Station NRCR11,0,,AUTOSTATS(Yes,,):
              Station NRCR12,0,,AUTOSTATS(Yes,,):
              Station TR2NR,0,,AUTOSTATS(Yes,,):
              Station NRCR21,0,,AUTOSTATS(Yes,,):
              Station NRCR22,0,,AUTOSTATS(Yes,,):
              Station RWR1,0,,AUTOSTATS(Yes,,):
              Station RWR2,0,,AUTOSTATS(Yes,,):
              Station TR1R,0,,AUTOSTATS(Yes,,):
              Station TR2R,0,,AUTOSTATS(Yes,,):
              Station TR3NR,0,,AUTOSTATS(Yes,,):
              Station WRTR1,0,,AUTOSTATS(Yes,,):
              Station WRTR2,0,,AUTOSTATS(Yes,,):
              Station CRNR1,0,,AUTOSTATS(Yes,,):
              Station WRTR3,0,,AUTOSTATS(Yes,,):
              Station CRNR2,0,,AUTOSTATS(Yes,,):
              Station WRTR4,0,,AUTOSTATS(Yes,,):
              Station WRTR5,0,,AUTOSTATS(Yes,,):
              Station WRTR6,0,,AUTOSTATS(Yes,,):
              Station TR3R,0,,AUTOSTATS(Yes,,):
              Station TR4R,0,,AUTOSTATS(Yes,,):
              Station ECRER1,0,,AUTOSTATS(Yes,,):
              Station ECRER2,0,,AUTOSTATS(Yes,,):
              Station TR1ECR,0,,AUTOSTATS(Yes,,):
              Station R,0,,AUTOSTATS(Yes,,):
              Station TR2ECR,0,,AUTOSTATS(Yes,,):
              Station CRDR1,0,,AUTOSTATS(Yes,,):
              Station TR4NR,0,,AUTOSTATS(Yes,,):
              Station CRDR2,0,,AUTOSTATS(Yes,,):
              Station TR5R,0,,AUTOSTATS(Yes,,):
              Station TR3ECR,0,,AUTOSTATS(Yes,,):
              Station TR4ECR,0,,AUTOSTATS(Yes,,):
              Station TR5ECR,0,,AUTOSTATS(Yes,,):
              Station NR,0,,AUTOSTATS(Yes,,):
              Station TR6ECR,0,,AUTOSTATS(Yes,,):
              Station TR6R,0,,AUTOSTATS(Yes,,):
              Station ERTR1,0,,AUTOSTATS(Yes,,):
              Station ERTR2,0,,AUTOSTATS(Yes,,):
              Station ERTR3,0,,AUTOSTATS(Yes,,):
              Station ERTR4,0,,AUTOSTATS(Yes,,):
              Station ERTR5,0,,AUTOSTATS(Yes,,):
              Station ERTR6,0,,AUTOSTATS(Yes,,):
```

```
Station DRCR11,0,,AUTOSTATS(Yes,,):
Station DRCR12,0,,AUTOSTATS(Yes,,):
Station TR1,0,,AUTOSTATS(Yes,,):
Station TR5NR,0,,AUTOSTATS(Yes,,):
Station TR2,0,,AUTOSTATS(Yes,,):
Station TR3,0,,AUTOSTATS(Yes,,):
Station TR4,0,,AUTOSTATS(Yes,,):
Station TR5,0,,AUTOSTATS(Yes,,):
Station TR6,0,,AUTOSTATS(Yes,,):
Station DRCR21,0,,AUTOSTATS(Yes,,):
Station DRCR22,0,,AUTOSTATS(Yes,,):
Station WRCR11,0,,AUTOSTATS(Yes,,):
Station WRCR12,0,,AUTOSTATS(Yes,,):
Station TR6NR,0,,AUTOSTATS(Yes,,):
Station WRCR21,0,,AUTOSTATS(Yes,,):
Station WRCR22,0,,AUTOSTATS(Yes,,):
Station TR1NR,0,,AUTOSTATS(Yes,,):
Station NRTR1,0,,AUTOSTATS(Yes,,):
Station CRWR1,0,,AUTOSTATS(Yes,,):
Station NRTR2,0,,AUTOSTATS(Yes,,):
Station CRWR2,0,,AUTOSTATS(Yes,,):
Station NRTR3,0,,AUTOSTATS(Yes,,):
Station NRTR4,0,,AUTOSTATS(Yes,,):
Station NRTR5,0,,AUTOSTATS(Yes,,):
Station NRTR6,0,,AUTOSTATS(Yes,,);
```