Automation in Construction 47 (2014) 37–49

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Evaluation of queuing systems for knowledge-based simulation of construction processes

Reza Akhavian [1], Amir H. Behzadan *

Department of Civil, Environmental, and Construction Engineering, University of Central Florida, 4000 Central Florida Blvd, Orlando, FL 32816-2450, USA

ARTICLE INFO

Article history:
Received 28 January 2014
Received in revised form 11 June 2014
Accepted 17 July 2014
Available online xxxx

Keywords:
Construction
Infrastructure
Simulation
Discrete event simulation
Queue
Queue discipline
Wireless data collection
Data mining
Knowledge
Ultra wideband

ABSTRACT

During the course of a construction project, there are many situations in which formation of waiting lines or queues is inevitable. The effect of resource delays in queues on the overall project completion time and cost has motivated researchers to employ simulation for analysis of queuing systems in order to identify the best operational strategies to reduce the time wasted in queues. Providing proper and timely input data with high spatial and temporal accuracy for queuing systems simulation enhances the reliability of decisions made based upon the simulation output. Hence, the presented paper describes a methodology for collecting and mining of spatio-temporal data corresponding to the interactions of queue entities to extract computer interpretable knowledge for simulation input modeling. The developed framework was validated using empirical datasets collected from a series of experiments. The extracted relevant knowledge from the queuing system entities was used to update corresponding simulation models.

1. Introduction

Waiting lines or queues exist in almost all industrial and manufacturing processes. In all such queuing systems, there are entities that need to be repetitively processed by other entity(s). The entity waiting in a line to receive service is called a *client* and the entity that processes clients is called a *server*. Similar to queuing systems in manufacturing settings, in many construction systems, clients (or resources) can be delayed in waiting lines when a server (or processor) is already captured by a previously arrived client and thus is busy.

A classic example of a construction queuing system is the arrival of dump trucks in a loading area where excavators or front end loaders load them with soil. As shown in Fig. 1, cyclic activities of an earthmoving operation consist of load, haul, dump, and return processes. A part of this cycle that embraces the waiting line and server is considered as the queuing system. Therefore, it is clear that the boundaries of the system are not necessarily spatially fixed and can dynamically change depending on the length of the queue and the efficiency of the server.

As soon as a client arrives inside the boundaries of the system, depending on the state of the server (i.e. idle or busy), it either waits in the queue or proceeds to be served immediately. Once the service is completed, the client leaves the system and its state, attributes, and other properties will no longer affect the conditions and properties of the queuing system. That is why in queuing systems terminology, the arrival of a client in the system is also referred to as the client's *birth* and its departure from the system is called the client's *death*, which imply that only the time that a client spends inside the queuing system is of interest to queuing analysis [1]. A final note on Fig. 1 is that although it shows a construction operation cycle, a queuing system may not be necessarily part of a cyclic operation; that is, the clients that enter the system may not return and the characteristics of the queuing system do not depend on the clients' identifications.

A construction manager who deals with an operation that involves queues is most often interested in knowing the waiting time during which a resource is delayed in a queue, the service time or how long it takes for the server to finish processing a specific client, and the logistics of the queue (i.e. number of delayed resources in a queue, or the queue length). Such knowledge is of critical importance to allocating the optimal number/type of resources, configuring the site layout, estimating the productivity, and determining the durations of individual operations as well as the entire project. Towards this goal, simulation models have been widely used in modeling queuing systems and to obtain valuable insight into the characteristics of the queues and their impacts on the overall project [2]. As previously stated, this is mainly due to the fact that the processing of clients by a server is a repetitive task and

* Corresponding author. Tel.: +1 407 823 2480; fax: +1 407 823 3315.
E-mail addresses: reza@knights.ucf.edu (R. Akhavian), amir.behzadan@ucf.edu (A.H. Behzadan).
[1] Tel.: +1 407 823 2480; fax: +1 407 823 3315.

http://dx.doi.org/10.1016/j.autcon.2014.07.007
0926-5805/

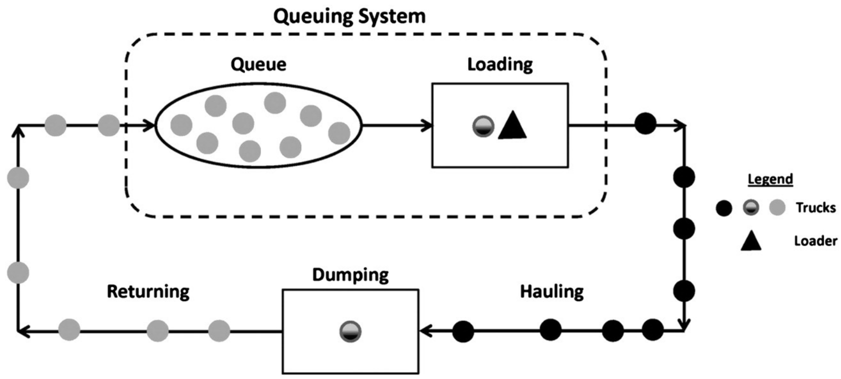

Fig. 1. Example of a single server queuing system in construction projects.

simulation models are perfect tools to predict the performance measures of repetitive processes of undeterministic nature. Among others, discrete event simulation (DES) models are particularly employed in construction and infrastructure projects since most often, the entire construction system can be broken down into discrete processes [3–5]. Within the construction and infrastructure domain, proper modeling of queuing systems is not a trivial task due to the stochastic, uncertain, and transient nature of such operations. A good example of such stochasticness that happens frequently and needs to be analyzed in the context of construction management is rework [6,7]. In queuing systems, in order to model the uncertainties in customer arrival times, most mathematical queuing theories suggest the use of specific probability distributions such as the exponential distribution [1,2,8]. However, previous research in construction systems based on real world observations of resource arrivals (e.g. dump trucks waiting in line to receive service from a front end loader or an excavator) indicated that the assumption of exponentially distributed arrival times can be often invalid [8–11]. Moreover, there are other important properties of a queuing system such as the queue discipline that must be accurately modeled when simulating queue operations. In particular, the sequence of queue operations not only can follow any of the well-known disciplines (which will be described in details in this paper), but also can be adjusted occasionally due to spatio-temporal requirements of the jobsite and the real vs. planned work progress [9,12]. Halpin and Riggs [9] indicated “breaks in queue discipline” as the first challenge among several difficulties in filed applications of queuing models. According to Martinez [12], discipline expression in modeling construction queues can be very dynamic and dependent on resource dynamic properties. Therefore, modeling of queuing systems requires accurate input with regard to queue properties that may change over the course of a construction project.

The necessity of providing a simulation model with accurate input data describing queuing systems and client–server interactions under dynamic and uncertain conditions highlights the importance of utilizing adaptive DES models that can be updated and fine-tuned in accordance to operations-level changes occurring in the real system. This requires meticulous data collection and mining processes to enable extracting relevant knowledge necessary to build the simulation model. To this end, this paper describes algorithms designed to extract knowledge pertinent to client–server interactions in queuing systems and to provide computer interpretable input for corresponding simulation models. First, a description of relevant previous studies is provided and identified gaps resulting in the presented research are discussed. Next, major properties that characterize a queuing system are introduced and their significance in designing construction and infrastructure simulation models is explained. Then, the algorithms that were designed and implemented to find and represent queue properties inside simulation models are described and the underlying mathematical background is briefly explained. Finally, the robustness and effectiveness of these algorithms will be examined using empirical data and results will be discussed.

2. Research background

Although utilizing operations-level simulation models that help achieve high levels of efficiency in managing construction projects has been explored and widely advocated in academic research [3,9,13,14], there is still much room for investigating their real value and potential applications that can result in their systematic accreditation by the construction industry [3,15]. Recent studies tried to investigate the reasons behind the limited and often, isolated use of simulation models in large scale by the industry. Among others, it was stated that most existing construction simulation systems rely on historical data and expert opinions to create simulation models [16]. Given the dynamics involved in most construction systems, such input data may turn out to be unrealistic (resulting in optimistic or pessimistic output), and are often hard to be independently verified. Therefore, the output of the resulting simulation models can be far from the realities of the operations on the ground. In the absence of methods that facilitate the process of constantly updating these simulation models with factual data from the real construction system, such models will soon be obsolete and of little (if any) value to the decision-making process [3,4,16]. In order to alleviate this problem, it has been previously discussed that collecting factual data as the project makes progress, discovering meaningful knowledge from these data, and feeding the extracted knowledge to corresponding simulation models can be a promising approach [17]. In order to achieve this, the possibility of collecting, fusing, and mining process data has been recently investigated by the authors through developing an integrated framework for construction equipment data-driven simulation models [17–19]. There have also been other sparse studies aimed at addressing this problem in limited scopes [20,21]. Despite these efforts, in almost all previous studies, project resources and entities were considered as single units for data collection and little knowledge was produced from the collected data to describe how individual entities would interact with one another at the process-level over time. AbouRizk et al. [15] indicated that the first requirement of developing a construction simulation model is acquiring knowledge about the logic and sequence of the operation. Knowing the interactions, relationships, and interdependencies between different entities is a crucial step in acquiring knowledge about the logic and sequence of activities, and can reveal potential predominant work patterns dictated by some entities. Therefore, in the context of queuing systems where entities are in constant interaction with one another, acquiring accurate data to generate knowledge pertaining to the client–server relationships is necessary for developing valid simulation models.

A number of researchers studied the implementation of queuing systems in construction simulation modeling. For instance, in one study, the FLEET program, queuing theory, and DES were used for selection of loader-truck fleets in infrastructure projects [10]. Using DES models, Ioannou [22] investigated the formation of queues during the process of rip-rap placement for the construction of a dam embankment. In another study, the probability distributions of mixer trucks' arrival and service times in concrete delivery and placement were examined and best fit probability distributions for activity durations were identified [23]. In addition, queuing input data uncertainty in earthwork projects was investigated using a probabilistic queuing model with fuzzy input and fuzzy probabilities and also a purely fuzzy queuing model [24]. However, to the authors' best knowledge, none of the aforementioned studies explored the potential of providing a DES model with factual data describing queuing systems in order to generate a more realistic simulation model. In the next section, major queue properties as relevant to the goals and discussions presented in this paper are presented.

3. Queue properties

Queues are characterized by a number of properties that represent the interrelationships between the entities involved in a queuing system. The arrival process, service duration, and queue discipline are among the most important properties of a queue [25]. In addition to these basic properties, the numbers of servers, capacity of the queue, and the population of entities to be served are some of the other properties of a queuing system. However, since information related to these latter properties are often provided as part of the project specifications (e.g. site layout and temporary route arrangement may dictate the number of dump trucks that can form a queue close to the loading area at any given time) or equipment manufacturers' catalogues (e.g. bucket capacity of an excavator can be used to determine how many dump trucks can be served within a certain time period), further onsite data collection and analysis regarding these properties do not contribute much to simulation model input data generation and thus are not the main focus of this study.

As previously discussed, queue properties and the ability to detect and use their exact and correct values are essential in designing complex DES models. For instance, *Nonstationary queues* – in which the complicities are due to the interactions that occur while the equipment are moving in traffic – in essence are subject to varying measures. In a construction of dam embankment with nonstationary queues, Ioannou [22] modeled a sophisticated operation using STROBOSCOPE in which activity durations and queue discipline depended on the operation's progress and certain measures over the course of the project. STROBOSCOPE is a programmable and extensible simulation system designed for modeling complex construction operations in detail and for the development of special-purpose simulation tools [12].

3.1. The arrival process

In a queuing system, the arrival process can be specified by a sequence of interarrival times that are independent and identically distributed (IID) as a simplifying assumption in order to fit probability distributions with fixed parameters. The randomness involved in the arrival occurrences makes it easy for the interarrival times to be characterized by a probability distribution [26]. For convenience, random arrivals are often modeled as a Poisson process with exponentially distributed interarrival times with a rate of λ (number of arrivals in unit time) and mean of $1/\lambda$ (average interarrival time) [1]. Despite its ease of use and widespread application in queue modeling, the exponential distribution may fail to fully reflect the real world observations taken from actual clients' arrivals and interarrival times especially in systems with transient and constantly changing states (e.g. construction projects) [8,11].

3.2. The service process

A service facility (i.e. where a server processes clients) can have different number of channels and phases. In the presence of more than one server, channels refer to available routes clients can take after having waited in line to reach the service facility [2]. An earthmoving operation with two excavators is an example of multi-server queuing system that has two channels. Phases are the number of stops a client must make after getting in line and before the service is completed. For instance, a stockpile of precast concrete segments may need to be moved by a tower crane to another place where a heavy lifter puts them on flatbed trailers. The service facility depicted in Fig. 1 is single-channel single-phase. In addition, the time it takes for clients to be served by server(s) (a.k.a. service time) is another determining factor in queuing systems. Similar to the interarrival times, service times are IID and can be represented by a probability distribution [26].

3.3. The queue discipline

Queue discipline is defined as a rule or set of rules based on a specific attribute of the entities. It determines the pattern (i.e. order) by which entities in the queue receive service [25]. The choice of the queue discipline and the rules to be applied can significantly affect the number of entities waiting in a queue, the average waiting time, and the efficiency of the service facility [27]. The most common queue discipline is first-in-first-out or FIFO in which clients in line are served based on their chronological order of arrival. Although FIFO has been long used as a default queue discipline in modeling queuing systems [25], in many scenarios, it is equally likely that clients be served according to other service patterns. For instance, the last-in-first-out or LIFO discipline may be the case in situations where a heap or stack of clients (e.g. raw materials, prefabricated concrete segments, steel sections) is waiting to be processed by a server. Other than FIFO and LIFO, the serving pattern of a queuing system can be characterized according to an intrinsic attribute of the entities in the system. This type of queue discipline is called *priority queues* or PRI queue discipline. In the example of a queue of dump trucks waiting to be loaded by an excavator, priority might be given to those dump trucks with less fuel left. Sometimes, there may be no rule according to which clients receive service from the server, in which case the queue discipline is considered as service-in-random-order or SIRO. Fig. 2 illustrates the concepts of FIFO, LIFO, and PRI queue disciplines. In this figure, clients are specified by letter C and the server is specified by letter S. Each of the three queues shows the client that should be drawn from the queue under the specified queue discipline.

Also, Fig. 3 summarizes the properties of the queuing system shown in Fig. 1 in each stage of the process.

Considering the diversity of queues, a notational system is widely used to distinguish between different systems. An abridged version of the notation is based on the $A/B/c/D$ format, in which A represents the interarrival probability distribution, B represents the service time probability distribution, c represents the number of parallel servers, and D represents the queue discipline [28]. For consistency, this notational system is used throughout this paper. Inside a simulation model, interarrival times and service times may be represented by probability distributions and queue discipline can be defined using specific functions provided by the DES platform. In the next section, the process of finding the best mathematical representations of interarrival times and service times will be discussed.

FIFO

LIFO

PRI

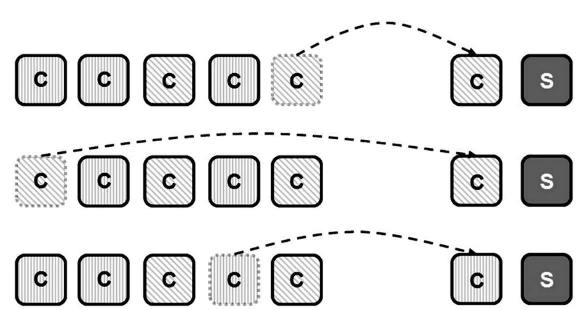

Fig. 2. Demonstration of FIFO, LIFO, and PRI queue disciplines.

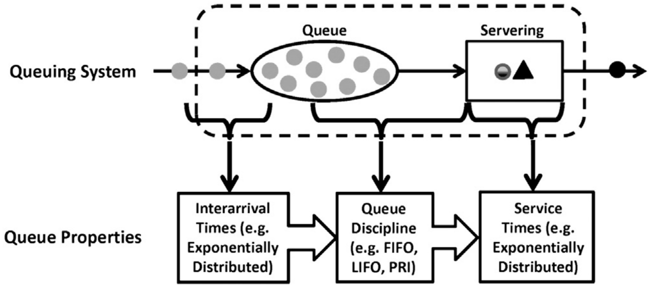

Fig. 3. Sequence and the properties of a queuing system.

4. Mathematical representation of interarrival and service times

In data-driven DES modeling, data necessary to describe the arrival process and service durations of the queuing system should be collected from the real world system. For the purpose of this research, details of data collection and knowledge extraction methodologies designed and already validated by the authors are described in [17]. Law and Kelton [11] stated that in case of having access to actual data on a certain input random variable, three approaches can be adopted to specify a distribution corresponding to the available data and use the distribution in a DES model accordingly. The first approach is to use one of the observed data values every time it is needed in the simulation. The second approach is defining an empirical distribution function based on the collected data and sample from that distribution whenever those input data are required in the model. Finally, the third approach is fitting a standard theoretical distribution to the collected data points. Generally, all the aforementioned methods have their uses in different applications and none can be ignored. However, in the context of DES modeling of construction and infrastructure systems, and considering the stochastic and uncertain nature of activities, the first approach will almost certainly result in a biased model with unrealistic representation of interarrival times or service durations. The second approach may seem to be viable for adoption at the first glance. However, it has some shortcomings that may make it unsuitable for the purpose of this study. In the context of this paper, in particular, since an empirical distribution function is just representing the values that exist in the pool of collected data, the quality of the distribution relies solely on the quality of the sample. As a result, there might be some irregularities in the distribution function. This causes even more problems when the number of collected data points is small. Moreover, an empirical distribution function can only generate values inside the range of the collected data. For example, there could be a few instances of very large service times that may have occurred before or after the data collection period, and thus will not be reflected in the empirical distribution function [11]. This is not desirable in designing a simulation model because the performance of a model depends significantly on the potential of predicting an extreme event that may not be part of the empirical data but the probability of which can still be captured at the tails of a theoretical probability distributions [11]. When a theoretical distribution is used, if it is extremely unlikely for a variable to exceed a certain value, say y, the distribution can be truncated at y to better represent the reality. Therefore, it is imperative that fitting a standard theoretical distribution to the collected data can better guarantee a more realistic sampling of the observed values inside a simulation model.

Collected data points and their corresponding probabilities can be shown by means of histograms [11]. In some cases, it is possible to find one or more probability distributions that best match a histogram. However, when dealing with more than one candidate distribution that looks representative, a closer evaluation of the level of fitness is necessary for accurate input modeling. A commonly used approach to systematically assess the quality of a fit is using goodness-of-fit tests. A goodness-of-fit test evaluates a null hypothesis, H_0, specifying that the random variable conforms to the assumed distribution function against the H_1 hypothesis that states the opposite [25,29]. The procedure consists of calculating a test statistic and comparing it with a critical value to see if it exceeds that value and thus, there is sufficient evidence to reject the null hypothesis.

There are three commonly used goodness-of-fit tests for evaluating the quality of fitness; 1) Chi-Square test, 2) Kolmogorov–Smirnov (K-S) test, 3) and Anderson–Darling (A-D) test. The goodness of this comparison is appraised based on the distribution of test statistic as shown in Eq. (1), which approaches the chi-square (χ_0^2) distribution with $k-1$ degrees of freedom [25].

$$\chi_0^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

This test is based on the null hypothesis of no considerable difference between the expected and sample distributions. The universal applicability of the Chi-Square test makes it a commonly used method, however, since the test depends on the number of intervals, it is usually advised to use this test in addition to other tests and with special care [11,25,29].

The Kolmogorov–Smirnov (K-S) test compares the experimental cumulative distribution function (CDF) with the CDF of an expected theoretical distribution [25]. The empirical CDF $S_N(x)$ is defined by Eq. (2),

$$S_N(x) = \frac{\textit{number of random } R_1, R_2, \ldots, R_N \leq x}{N} \tag{2}$$

The K-S test is based on the largest absolute deviation (D) between $F(x)$, the CDF of the theoretical distribution and $S_N(x)$, which is based on test statistic [25] as shown in Eq. (3),

$$D = \max |F(x) - S_N(x)| \tag{3}$$

Both K-S and A-D tests directly compare the hypothesized model's distribution to the empirical distribution; however A-D places more emphasis on the differences found in the tails. The test statistic for A-D is defined by Eq. (4),

$$A^2 = -n - \frac{1}{n} \sum_{i=1}^{n} (2i-1)\left[\ln(F_x(x_i)) + \ln(1 - F_x(x_{n+1-i}))\right] \tag{4}$$

in which x is the random variable, n is the number of the data points, and A^2 is the test statistic. It is generally accepted that the A-D test provides better results than the K-S and Chi-Square tests [11,29].

All three goodness-of-fit tests have been previously applied to construction simulation problems [23,30,31]. However, despite its better performance, the A-D test was not extensively used in construction simulation modeling [32]. In this study, all three tests are employed to find the best fit probability distributions.

5. Queue property discovery algorithms

5.1. Interarrival and service times

In this research, data that are collected using multimodal data collection sensors include, among others, the coordinates of each tracked entity (i.e. clients and servers) identified by its unique ID number at any point in time. This data collection and storage technique creates a pool of unique data points that contains ID numbers attributed by coordinates x_t^{ID}, y_t^{ID}, z_t^{ID} that indicate the longitude, latitude, and altitude of each entity (the current discussion only concerns the position of entities in 2D, but the third dimension is also collected and can be useful in some cases). As soon as an entity enters the boundaries of the queuing system, its arrival time, t_a, is marked and stored. Depending on the availability of the server, it then either enters the queue or proceeds directly to the server. More detailed discussion about the reasoning process deployed to discover the exact state of an entity in the system and all modes of data contributing to the knowledge discovery phase can be found in [17]. In a nutshell, the reasoning process is a taxonomy-based approach that first finds the state of construction resources, and then, according to the stream of multi-modal data extracts the contextual process knowledge. If the entity joins the queue, its waiting time is calculated by subtracting the time it starts receiving service, t_s, from t_a. If no queue was formed and the server is free waiting for the next client, there is no waiting time associated with this client in the queuing system and thus $t_s - t_a = 0$. Depending on the service duration, the client leaves the server at t_f when its processing by the server is finished, and the duration of the service can be calculated as $t_f - t_s$. Therefore, the timing status of the client in the system can be represented as *ID* (t_a, t_s, t_f). When the service starts, the coordinates of the client should be close to those of the server. This condition confirms that the client is in fact in the process of receiving service at that moment. For example, for the client 020 (09 : 46 : 16, 09 : 47 : 03, 09 : 48 : 35) which is served by server ID 1, the following conditions must be met at the time of service,

$$x^{20}_{t_s} \cong x^{1}_{t_s} \quad , \quad y^{20}_{t_s} \cong y^{1}_{t_s} \tag{5}$$

Considering such spatio-temporal relationships between clients and server(s), the waiting time and service time can be obtained. The interarrival time of the queuing system can be also calculated by taking into account the arrival pattern of all clients over a specific period of time (i.e. duration of real system observation). In particular, the difference between the arrival times of any two subsequent clients can be stored as a new interarrival time value.

5.2. Queue disciplines

As discussed earlier, there are a variety of rules that determine the order according to which clients are drawn from a queue for further processing by the server. Compared to the FIFO and LIFO situations that are the two most common queue disciplines, PRI queues introduce more complexity in terms of the interaction between clients and server(s) and thus, require more investigations [33,34]. In this section, the methodology designed for discovering all such queue disciplines from the incoming client–server data streams is described.

As previously stated, clients' arrival times (t_a) and service start times (t_s) can be extracted by capturing positional data. Having obtained data from several service instances, the chronological record of collected t_a and t_s values is used to create and populate two separate lists hereafter referred to as the *Order of Arrivals (OA)* and the *Order of Services (OS)*. These two lists serve as the backbone of all algorithms described in this section. In essence, knowledge revealing a specific queue discipline is discovered through a side-by-side analysis and comparison of entries and their chronological orders in these two lists. The general algorithm for discovering the true queue discipline is represented in the flowchart shown in Fig. 4.

Following the flowchart of Fig. 4 and given an *OA* list, each of the three sub-algorithms (corresponding to FIFO, LIFO, or PRI disciplines) first creates an *Expected Order of Service (EOS)* list. The generated *EOS* list is then compared to the actual *OS* list (as generated from the real data). Depending on the degree of compatibility (i.e. similarity) between the *EOS* and *OS* lists, each sub-algorithm outputs an accuracy level by which the collected data from the real system resemble the queue discipline represented by that sub-algorithm.

5.2.1. First-in-first-out (FIFO) queues

FIFO is often considered as the default queue discipline in most simulation software [25]. Discovering a FIFO discipline by analyzing the *OA* and *OS* lists is relatively simple, as illustrated in the flowchart of Fig. 5.

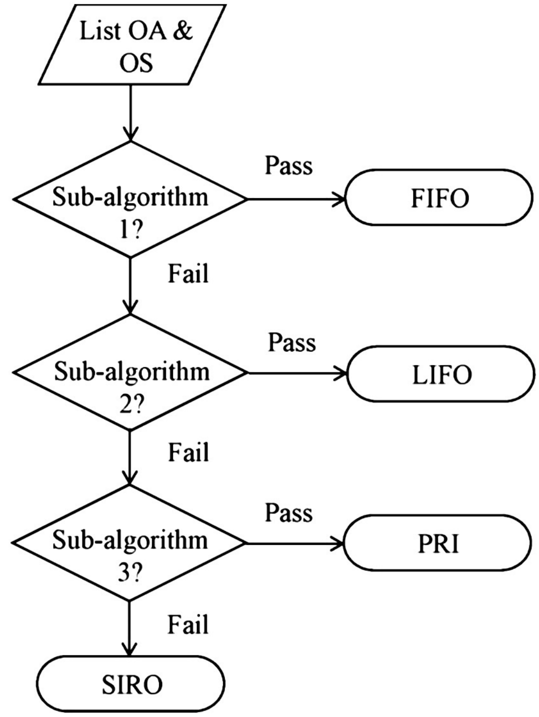

Fig. 4. Main flowchart for queue discipline discovery.

According to this figure, the *EOS* list in this case should be identical to the *OA* list as the clients receive service in the same order they arrive in the queue. Therefore, the algorithm verifies if the actual *OS* list is compatible with the *EOS* list. If each and every element (i.e. client IDs) in the *EOS* list is equal to the corresponding element in the *OS* list, FIFO is the queue discipline (with 100% accuracy level) that best describes the client–server interactions represented by the collected data.

5.2.2. Last-in-first-out (LIFO) queues

For a queue with LIFO discipline, drawing a conclusion is more involved than the previous case. The rule states that the entity that arrives last should be served first. The main complexity in this case lies in the timing of when a server becomes available (i.e. giving service to the last client has just ended) and the timing of clients arriving in the queue, which may result in scenarios in which the *EOS* list is almost never generated by inverting the *OA* list. In particular and given a queue formation with n clients and a free server, the server must process the last client (i.e. client n) first. While processing client n, if a new client arrives in the queue (with $n - 1$ clients left) then the next client to be served is always the very last client in the queue (i.e. new client n). However, if no new client arrives in the queue between two

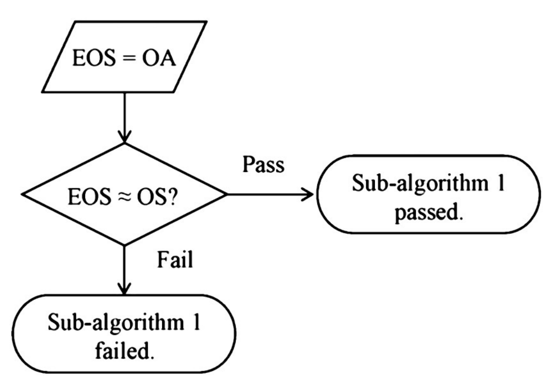

Fig. 5. FIFO queue discipline discovery flowchart.

instances of server availability, then the second to last last client (i.e. client $n-1$) will be served next since the very last client (i.e. client n) has just been served by the server. In general, if no new client arrives in the queue between k instances of server availability, then the next client to be served will be the kth last client ($k-1$ clients have already been served by the server).

Fig. 6 shows the flowchart of the LIFO discipline sub-algorithm. As shown in this figure, once the next available service time is read from the *OS* list, the client that arrives last in the queue is spotted. If the ID of this client is not among those who have been already served, it will be added to the *EOS* list as the next client to be served. Otherwise, it is concluded that no new arrivals have occurred in between two consecutive server availability instances, and thus the algorithm looks for the client who arrived one before last. This iteration continues until there is no more service time available and all clients have been already processed by the server.

5.2.3. Priority (PRI) queues

Priority queues are slightly different from queues with FIFO and LIFO disciplines since an external rule determines the order by which clients receive service. The external rule is often defined such that it assigns higher priorities to some clients over others based upon an attribute intrinsic to clients (e.g. size, type) or combinations of attributes [34]. A FIFO or LIFO queue is in fact a special case of a PRI queue in that the attribute used to prioritize the clients is the time of arrival in the queue. In its simplest form, an algorithmic approach for discovering the PRI discipline in a queuing system starts with considering two channels for the clients that arrive in the queue. In practice, this resembles the case where clients with higher service priority are channeled through a designated line while all other clients are lined up in a regular line. A common example of such channels is the passenger boarding process at an airport or bus terminal, where passengers whose boarding passes show "premier access" can board at any time of their choice using a special lane [35]. In this research, and in order to design a generic approach to discover queue discipline in PRI queues, the notion of two physically-separated channels is, however considered secondary. In other words, it is not necessary for clients to form two separate lines; the sub-algorithm developed for this queue discipline can properly function even if all clients are placed in a single line and in any arbitrary order after entering the boundaries of the queuing system.

Fig. 7 illustrates a priority queuing system in the moment the server has just become available (i.e. the client in service has just left the boundaries of the queuing system). Here, clients that are sorted in the ellipse based on their t_a (smaller t_a is placed closer to the server) are of two types: square and circle. Let's assume that squares have higher priority over the circles. Therefore, they can be grouped into two imaginary lines: a priority line and a regular line. Let's also assume that each of these two imaginary lines follows a FIFO discipline, internally. Essentially, before any client is drawn from the regular line for processing, the designed algorithm constantly checks whether there is any client waiting in the priority line. Therefore, in the queuing system of Fig. 7, assuming that during the next four server availability instances, no new client is added to the queue, all existing clients can be enumerated according to the order by which they will receive service. Clients of type square receive lower service numbers (higher priority) and clients of type circle receive higher service numbers (lower priority).

Fig. 8 shows the flowchart of the PRI discipline sub-algorithm. It should be noted that based on the general flowchart presented in Fig. 4, generated *OA* and *OS* lists are processed by all three sub-algorithms for all queue disciplines. As previously stated, in each case, an accuracy level will be calculated for every queue discipline (i.e. FIFO, LIFO, and PRI) based on the degree of compatibility between the *OS* and *EOS* lists. The queue discipline with the highest accuracy level can then be used as the dominant queue discipline when simulating the real system. Consequently, if none of the queue disciplines has a high enough accuracy level, it can be inferred that no specific rule was used to draw clients for processing. In this case, the SIRO discipline can be used to describe the queuing mechanism in the corresponding simulation model. In the following section, a comprehensive queuing scenario will be presented to better describe how raw collected data are transformed and used by the queue discipline discovery algorithm, and how accuracy levels are calculated for each potential queue discipline.

6. Empirical analysis and validation of results

In order to evaluate the functionality of the designed queue property discovery algorithms in extracting key queue parameters (i.e. interarrival times, service times, and queue discipline), a series of experiments was conducted in the Decision Support, Information Modeling,

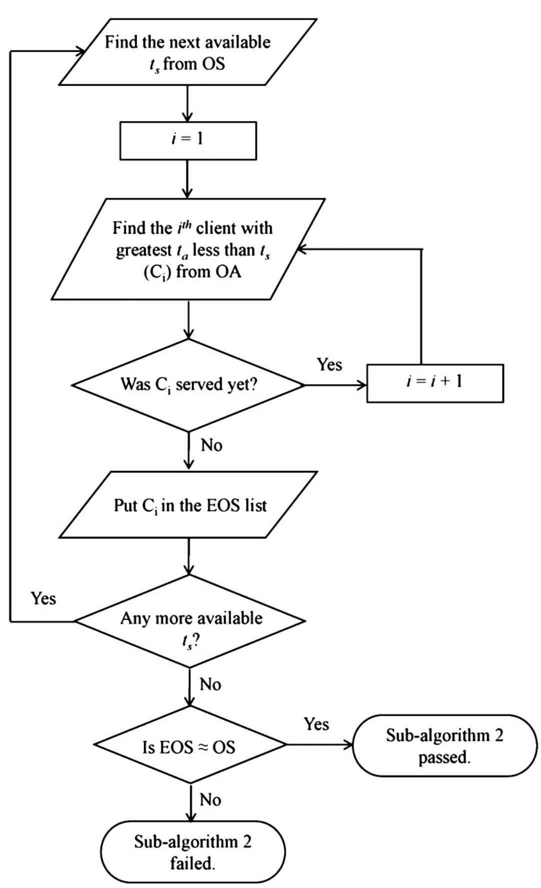

Fig. 6. LIFO queue discipline discovery flowchart.

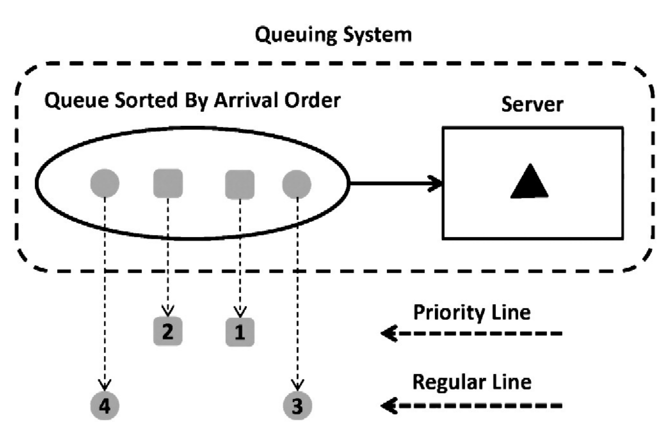

Fig. 7. Assigning priorities to clients in a PRI queuing system.

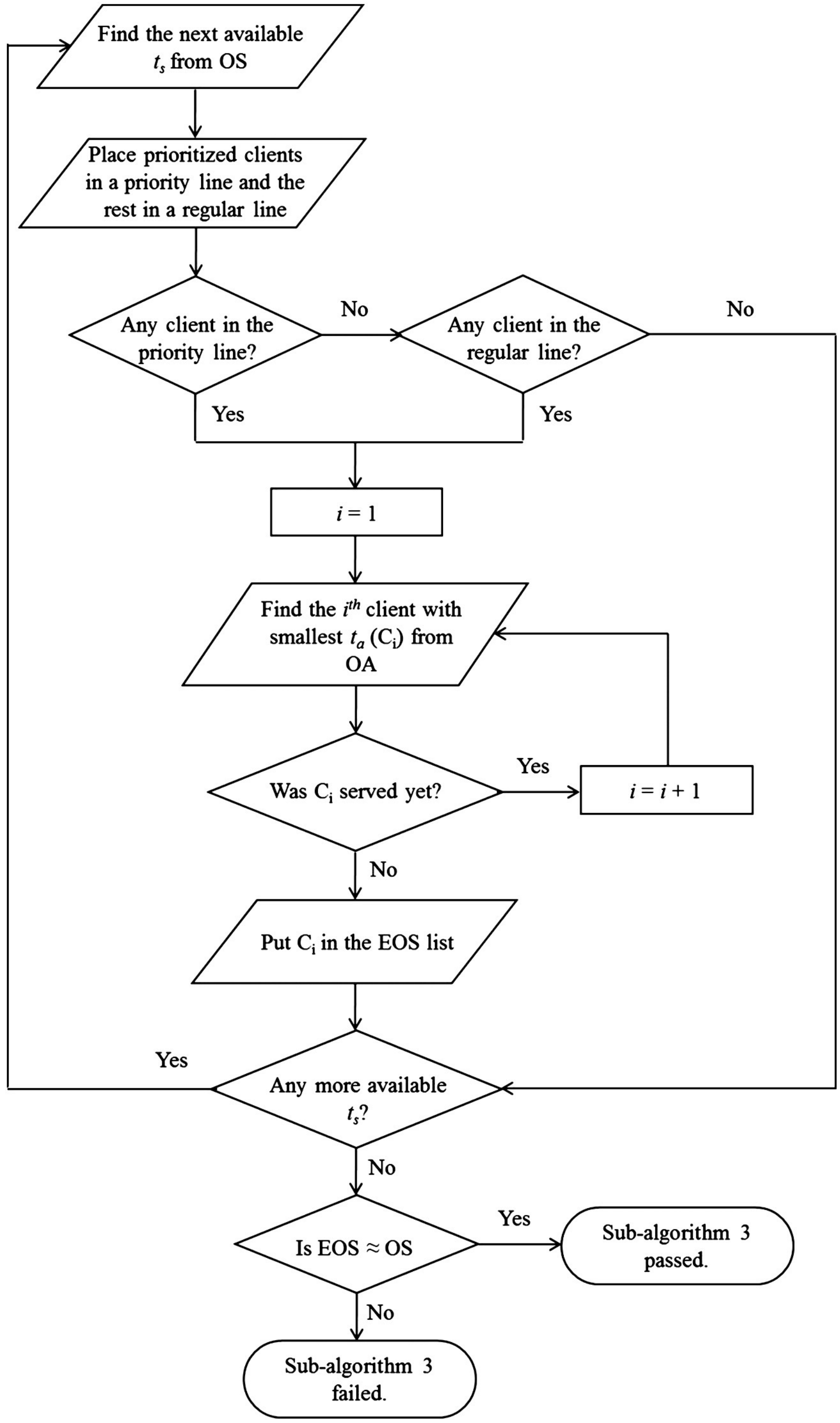

Fig. 8. PRI queue discipline discovery flowchart.

and Automation Laboratory (DESIMAL) at the University of Central Florida. In each experiment, client–server interaction data were collected from the real system under a prescribed queue discipline, and the performance of the designed algorithms in correctly identifying the queue discipline as well as other queue parameters from the incoming data streams was examined.

Several experiments were conducted using commercially available ultra-wideband (UWB) receivers and tags that were continuously moving in a 3D space to represent clients arriving at, waiting in, and leaving a queuing system, as well as server(s) in charge of processing clients. The location of each tag was recorded in real time with an accuracy of 15 cm in 3D (XYZ) space. The update rate of the sensors was set to 16 Hz and a total of 11 tags were employed in each experiments. Out of the 11 tags, 10 were used to model clients and 1 was used as a server. In total, more than 78,000 time-stamped positional data points were collected.

Table 1 shows a sample chronological record of the data points collected in a two-second time interval in one of the experiments. The first column in this table shows the current time in hh:mm:ss format, the second column is the ID of the UWB tag, the third, fourth, and fifth columns contain the 3D coordinates of the tag, and the last column is the tag status. As soon as a tag entered the boundaries of the queuing

Table 1
Sample of the collected data points.

Time	Tag ID	X	Y	Z	Tag Status
18:28:06	020-000-154-195	3.91023	3.04932	0.15026	In System
18:28:06	020-000-147-220	3.85169	3.04145	0.149724	In System
18:28:06	020-000-147-212	3.08748	3.02152	0.178556	In System
18:28:07	020-000-147-252	3.62637	3.0435	0.265829	In System
18:28:07	020-000-147-020	3.89411	3.06027	0.132566	In System
18:28:07	020-000-147-207	3.87354	3.06197	0.148414	In System
18:28:07	020-000-147-175	3.79864	3.08372	0.13733	In System
18:28:07	020-000-147-211	3.706	3.06462	0.118912	In System
18:28:07	020-000-154-195	3.90496	3.04561	0.1512	In System
18:28:07	020-000-148-011	4.65458	3.95913	0.316904	Just Born
18:28:07	020-000-147-220	3.8502	3.04176	0.155713	In System
18:28:07	020-000-147-212	3.08696	3.02207	0.179014	In System
18:28:07	020-000-147-252	3.62177	3.04439	0.265192	In System
18:28:07	020-000-147-020	3.89573	3.06664	0.131962	In System
18:28:07	020-000-147-207	3.87055	3.06544	0.151006	In System
18:28:07	020-000-147-175	3.80224	3.08201	0.13583	In System
18:28:07	020-000-147-211	3.70344	3.06286	0.120768	In System
18:28:07	020-000-154-195	3.90306	3.04175	0.156936	In System

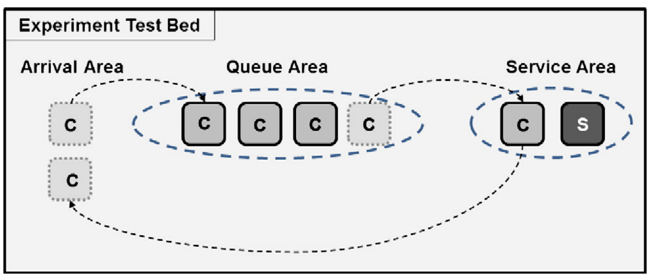

Fig. 9. Schematic illustration of the experiment test bed.

system, its status was marked as "Just Born" (e.g. tag 020-000-148-011 in Table 1). Otherwise, if it already existed in the queuing system, its status was shown as "In System".

Fig. 9 shows a schematic illustration of the experiment test bed. In this test bed, the queue area that hosts the 5 cm by 5 cm UWB tags (specified by C letters) is separated by 30 cm from the service area where another UWB tag acts as a server (specified by S letter).

In each scenario, a series of 50–60 instances of arrival (birth), service, and departure (death) was examined by human experiment assistants. Interarrival and service times were determined and scaled by observing the same number of instances in a real world earthmoving operation in which dump trucks waiting in a queue received service from a single excavator. More specifically, an assistant who was tasked with organizing the arrivals of entities used a stopwatch to add each entity to the queue area according to the scaled interarrival times observed in the real world earthmoving operation. Another assistant was responsible for drawing entities from the queue and putting them adjacent to the server tag. Again, the timing of this task was according to the scaled service durations taken from the real world earthmoving operation. Subsequent time-stamped arrivals of the client tags to the queue area determine the interarrival times and the time during which a client tag is in the service area (closest to the server) determines the service duration. Movement of tags does not affect the sensing accuracy.

Initially, three distinct experiments were conducted with FIFO, LIFO, and PRI queues, respectively. In each case, order of arrivals and services, and interarrival and service times were extracted using the methodology and algorithms described earlier. Moreover, the queue discipline in each case was discovered with a satisfactory accuracy level. Furthermore, once the performance of the algorithms in extracting the true queue discipline was verified, a fourth experiment was conducted in which the queue discipline was changed in the middle of the experiment to evaluate the performance of the designed algorithms in detecting the change as it occurred in the real system. A good example of why this experiment is relevant to scenarios from the real world is the case where a project manager, due to a variety of reasons (e.g. saving fuel, minimizing engine wear), decides to change a regular FIFO queue discipline to a PRI discipline that is based on the *fuel left* attribute of the dump trucks waiting to be loaded by an excavator [12]. Knowing the exact point in time when this queue discipline change occurs in the real system can help a modeler update the corresponding simulation model accordingly to better reflect the actual conditions on the ground, thus resulting in more accurate simulation output and more realistic performance prediction. In the designed experiment, the implication of the priority queue was such that UWB tags with even tag IDs were prioritized over the ones with odd tag IDs.

Table 2 shows a sample of the *OA* and *OS* lists along with the corresponding interarrival and service times extracted in one of the initial three experiments. Note that this table only contains a small portion (three-minute time interval) of a much larger record of data points collected during the experiment.

In each experiment, up to six commonly used standard theoretical probability distributions (i.e. Beta, Erlang, Exponential, Gamma, Normal, and Triangular) were fitted to the interarrival and service times. In each case, the quality of the fit was evaluated using the three goodness-of-fit tests (previously described) in the @RISK probability distribution fitting software. Next, probability distributions were ranked among each other according to their test statistics. Finally, out of the six representative probability distributions, the one that received the highest average rank was selected to be used inside the simulation model to describe the undeterministic characteristics of the queuing system. For example, Table 3 shows the results obtained for the first of the initial three experiments.

As shown in Table 3, for the first experiment, both the Gamma and Erlang distributions received the highest rank among all distributions

Table 2
Sample of the OA, OS, interarrival, and service times extracted in experiment 1.

Arrival Order	Arrival Times	Interarrival Times	Service Order	Service Start	Service Finish	Service Duration
020-000-147-020	18:25:38	0:00:17	020-000-147-020	18:27:27	18:27:39	0:00:12
020-000-154-195	18:25:55	0:00:21	020-000-154-195	18:27:42	18:27:47	0:00:05
020-000-148-011	18:26:16	0:00:19	020-000-148-011	18:27:50	18:27:54	0:00:04
020-000-147-212	18:26:35	0:00:07	020-000-147-212	18:27:57	18:28:17	0:00:20
020-000-147-252	18:26:42	0:00:07	020-000-147-252	18:28:20	18:28:26	0:00:06
020-000-147-211	18:26:49	0:00:32	020-000-147-211	18:28:30	18:28:37	0:00:07
020-000-147-175	18:27:21	0:00:08	020-000-147-175	18:29:01	18:29:11	0:00:10
020-000-147-220	18:27:29	0:00:07	020-000-147-220	18:29:14	18:29:27	0:00:13
020-000-147-207	18:27:36	0:00:15	020-000-147-207	18:29:30	18:29:48	0:00:18
020-000-147-020	18:27:51	0:00:05	020-000-147-020	18:29:52	18:30:00	0:00:08
020-000-154-195	18:27:56	0:00:11	020-000-154-195	18:30:03	18:30:17	0:00:14
020-000-148-011	18:28:07	0:00:28	020-000-148-011	18:30:21	18:30:35	0:00:14
020-000-147-212	18:28:35	0:00:12	020-000-147-212	18:30:39	18:30:42	0:00:03

Table 3
Ranking of the best fitted probability distributions for experiment 1.

	Test	Beta	Erlang	Exponential	Gamma	Normal	Triangular
Interarrival Time	*Chi-Square*	5	1	2	4	6	3
	K–S	1	3	4	2	5	6
	A–D	2	3	4	1	5	6
Rank		8	7	10	7	16	15
Service Time	*Chi-Square*	3	N/A	4	N/A	1	2
	K–S	3	N/A	4	N/A	1	2
	A–D	2	N/A	4	N/A	1	3
Rank		8	N/A	12	N/A	3	7

that were fitted to interarrival times and thus, either of the two was deemed suitable for use in the simulation model. Also, for service times, the Normal distribution had the best relative ranking among others. Moreover, the test statistics for both the Gamma/Erlang and Normal distributions were smaller than the critical value at 95% accuracy level in all three tests which imply that there is no reason to reject the H_0 hypothesis (i.e. there is no reason to reject the assumed distribution). In Table 3, the term N/A indicates that for that particular distribution, the data convergence failed and thus the distribution could not be fitted to the data. Fig. 10 shows the Gamma and Normal distributions as fitted to the extracted interarrival and service times in experiment 1, respectively. It should be noted that, although Normal distribution was selected as the best fit distribution, when using it inside a simulation model, appropriate lower and upper bounds should be defined, so as not to sample extremely low or large values. This can be done by using a truncated distribution with the same main parameters or adding control statements in the simulation script to ignore sample values smaller or larger than user-specified bounds.

The same procedure was followed for the next two experiments. Table 4 lists the best fitted distribution for the interarrival and service times for the initial three experiments. Note that all of the selected distributions as summarized in Table 4 had test statistics less than the critical value in all three goodness-of-fit tests which imply that in none of them can the H_0 hypothesis be rejected. In addition, in order to validate the selected probability distributions, the cumulative distribution function (CDF) of the actual empirical distribution constructed from the experimental data was compared against the final distribution selected using the two-sample K-S test statistic. Results are shown in the fourth column of Table 4, and the CDF graphs for the interarrival and service times of the first experiment are illustrated in Fig. 11. The two-sample K-S test used to compare the empirical data and the selected distribution should not be confused with the one-sample K-S test in Table 3

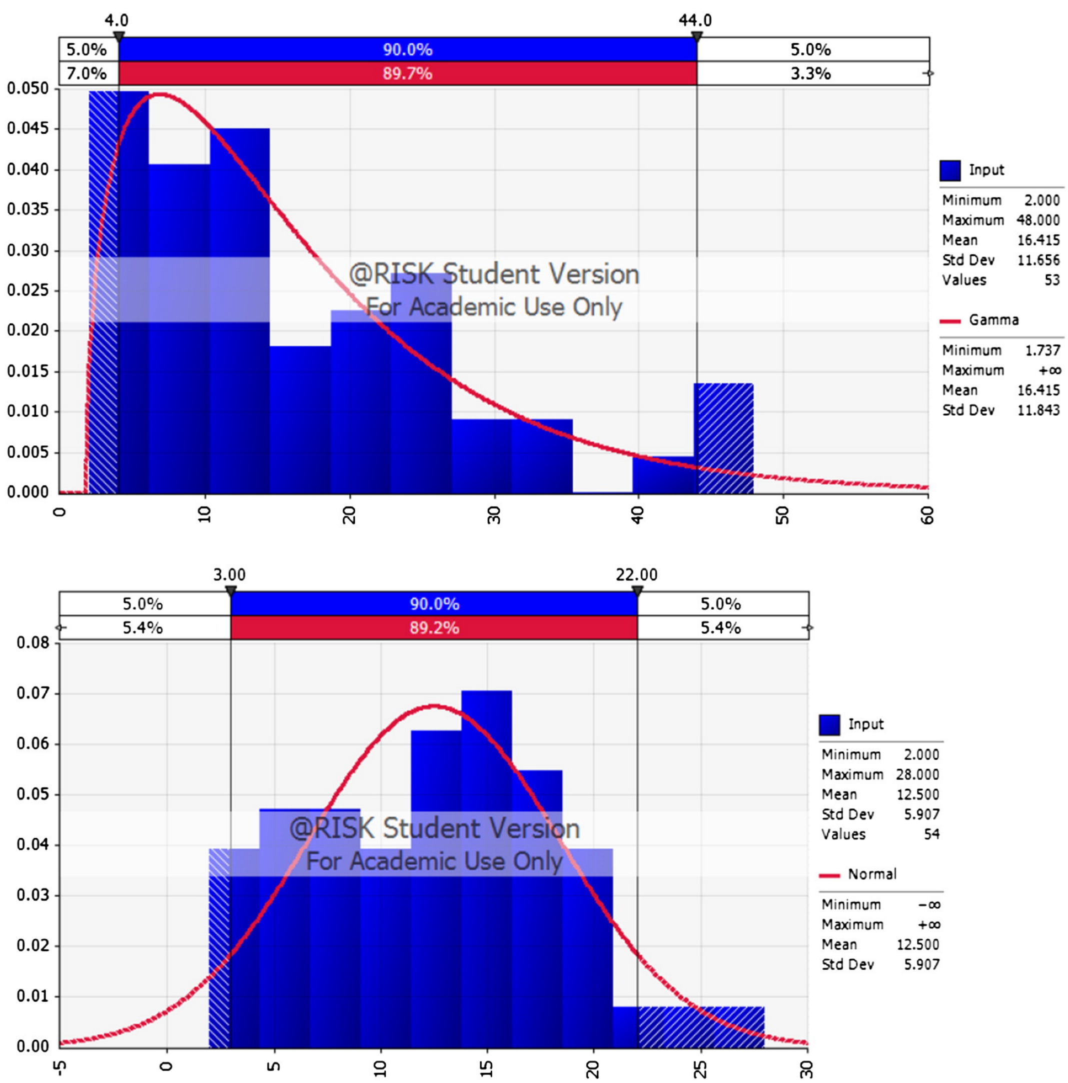

Fig. 10. Gamma and normal distributions fitted to the extracted interarrival and serivce times in experiment 1.

Table 4
Best fit distribution for experiments 1, 2, and 3.

Experiment	Queue Parameter	Best Fitted Distribution	Two-Sample K-S Test Statistic
1	Interarrival Times	Gamma [1.5,9.5]	0.09
	Service Times	Normal [12.5, 5.9]	0.06
2	Interarrival Times	Gamma [1.7, 8.1]	0.07
	Service Times	Beta [8.0, 3.5]	0.06
3	Interarrival Times	Erlang [2, 6.8]	0.16
	Service Times	Normal [16.16, 6.2]	0.11

that was used as a goodness-of-fit measure. The critical value of the two-sample K-S test for the level of significance of $\alpha = 0.05$ is given by Eq. (6),

$$D_\alpha = 1.36\sqrt{\frac{n_1 + n_2}{n_1 n_2}} \tag{6}$$

in which $\mathbf{D}_\alpha$ is the critical value at $\alpha = 0.05$, and $\mathbf{n}_1$ and $\mathbf{n}_2$ are the sizes of the samples [36]. Here $\mathbf{n}_1 \approx \mathbf{n}_2 = 55$ and thus $\mathbf{D}_\alpha \approx 0.26$. According to Table 4, all the test statistics are less than this critical value and thus the null hypothesis (samples are from the same distribution) is not rejected.

Once probability distributions were fitted to the interarrival and service times, the robustness of the designed methodology in discovering the true queue discipline was evaluated. As mentioned earlier, in the initial three experiments, FIFO, LIFO, and PRI queue disciplines were prescribed to the real system. Once time-stamped client–server positional data were collected, the *OA* and *OS* lists were generated and inputted to the queue discipline discovery algorithm illustrated in Fig. 4. Results indicated that the queue discipline of the first experiment (which was set to be FIFO in the real system) was discovered to be FIFO with 100% accuracy level. In essence, when *OA* and *OS* lists of this experiment were processed by sub-algorithm 1 in Fig. 4, it was observed that in 100% of the times the queue discipline was in fact FIFO. Also, the accuracy levels calculated by sub-algorithms 2 and 3 for the same *OA* and *OS* lists of experiment 1 were 5.60% and 8.30%, respectively. The results obtained from the queue discipline discovery algorithm for all initial three experiments are summarized in Table 5.

As shown in Table 5, all three sub-algorithms were successful in detecting the correct queue discipline that was adopted in each experiment. Also, in each case they detected the other two queue disciplines with relatively very low accuracy levels. Therefore, in order to draw a fair conclusion about the performance of the algorithms in each experiment, the last column represents the normalized accuracy levels considering all the results obtained in all sub-algorithms.

Finally, in the last (i.e. fourth) experiment, a change in the queue discipline was implemented at an arbitrary point in time during the experiment. In particular, while the queuing system was originally set to function under the FIFO rule, the discipline was switched to PRI when 50% of the service instances were fulfilled. Under the new queue discipline, the UWB tags with even tag IDs received higher priority over those with odd tag IDs. It was detected that the interarrival times and service times followed Gamma distribution and Normal distribution, respectively. Also, results of the data mining process indicated that a FIFO discipline was predominant in 54.7% of service instances and a PRI discipline was detected in 61.3% of service instances. Through normalizing the results, the output of the algorithm revealed that in the first 47.15% of service instances, a FIFO discipline was followed (compared to 50% as observed in the real experiment) while in the remaining 52.85% of service instances, the dominant queue discipline was PRI.

Results obtained from experiment 4 were imported into the STROBOSCOPE simulation system. Table 6 shows the input parameters of the simulation model based on the extracted interarrival and service times as well as the dynamic queue discipline, as extracted from the data collected from the client–server interactions in the real system.

The queuing system was modeled based on the activity cycle diagram (ACD) shown in Fig. 12. The Gamma distribution for the interarrival times was used to model the duration of the `Arrival` activity, and the Normal Distribution was used to model the duration of the `Service` activity. Also, the queue discipline was used to draw clients from the `ClientsWait` queue. Fig. 13 shows the STROBOSCOPE simulation script in which the implementation of the discovered queue discipline change in the model is highlighted (shown in front of the DESCIPLINE statement).

In order to check if providing the simulation model with the knowledge describing the exact timing of the queue discipline change makes a meaningful difference in simulation results, the mean waiting times of the two client types (i.e. tags with even and odd IDs) in the queue

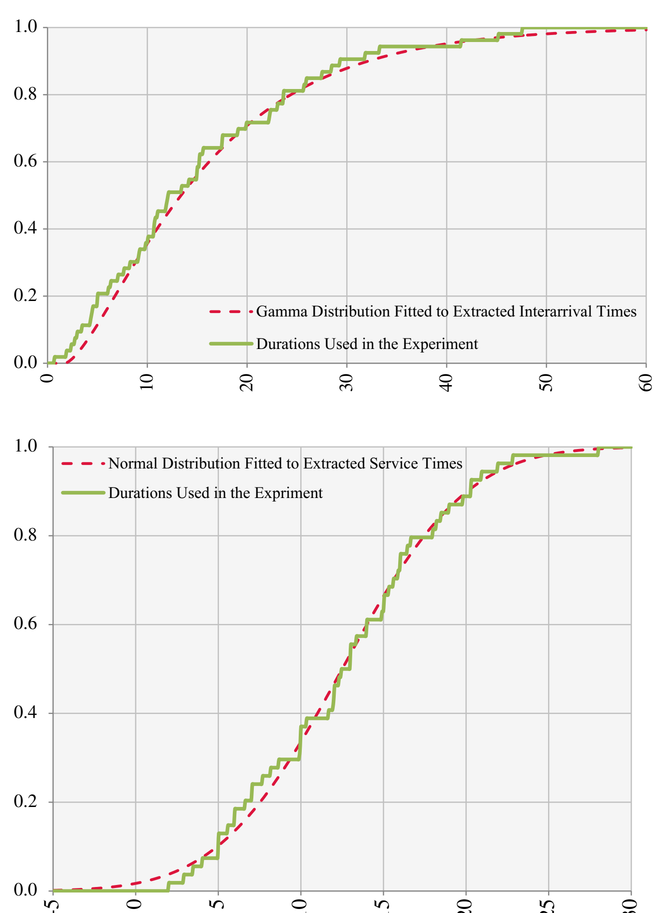

Fig. 11. CDF of the experiment input data and the best fit distribution to the extracted durations.

Table 5
Results of the queue discipline discovery.

Actual queue discipline (real world)	Calculated confidence level for the discovered discipline			Normalized confidence interval
	Sub-algorithm 1 (FIFO)	*Sub-algorithm 2 (LIFO)*	*Sub-algorithm 3(PRI)*	
FIFO	100%	7.54%	8.37%	86.32%
LIFO	16.66%	94.80%	2.21%	83.39%
PRI	16.98%	9.43%	97.23%	78.64%

Table 6
Input parameters of the simulation model for experiment 4.

Queue Parameter	Simulation Input
Interarrival times distribution	1.475 + Gamma (1.75, 8.10)
Service times distribution	Normal (13.66, 8.02)
Queue discipline	FIFO (47.15%) + PRI (52.85%)

were monitored in the simulation output. Eleven independent simulation models (each with 10 replications) were run each with a 10% increment in the percentage of service instances after which the queue discipline change occurred. In other words, in the first model, it was assumed that the queue discipline was changed from FIFO to PRI after 0% of service instances were completed (i.e. PRI discipline from the very beginning); in the second model, it was assumed that the queue discipline was changed from FIFO to PRI after 10% of service instances were completed; and eventually, in the eleventh model, it was assumed that the queue discipline was changed from FIFO to PRI after 100% of service instances were completed (i.e. FIFO discipline until the end). Each simulation replication was continued until 10,000 service instances were completed and therefore, a total of 100,000 service instances were modeled for each combination of FIFO and PRI percentages. As described earlier, the designed queue data mining algorithm revealed that in the first approximately 47% of the service instances the queue discipline was FIFO which was then switched to PRI. Therefore, one more simulation model (with 10 replications) was run in which the queue discipline was changed from FIFO to PRI after exactly 47% of service instances were completed. Fig. 12 shows the average waiting times for even ID tags and odd ID tags in seconds, obtained from these simulation models. In this figure, the horizontal axis shows the percentage of service instances in each simulation model after which the queue discipline is changed from FIFO to PRI, and the vertical axis indicates the average of the mean waiting times obtained from the results of 10 simulation

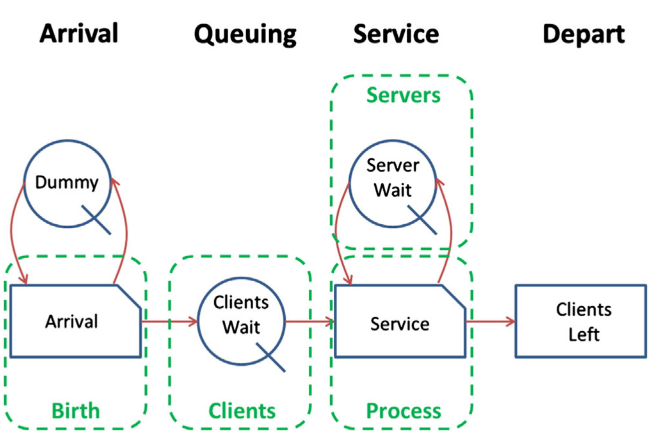

Fig. 12. Activity cycle diagram of the queuing system in STROBOSCOPE.

replications. The two points corresponding to the results of the experiment with 47% FIFO and 53% PRI are distinctively marked in the plots. As shown in Fig. 14, if the queue discipline for the entire process is PRI (i.e. 0% on the horizontal axis), the average waiting time for even ID tags (i.e. that have higher priority) is around 12 s, which is much less than the same value for the odd ID tags (around 81 s). At 47% on the horizontal axis (i.e. the time in the real experiment where queue discipline was changed from FIFO to PRI), the average waiting time for even ID tags (i.e. that have higher priority) is around 25 s and the same value for odd ID tags is around 63 s. Also, the average waiting time for all tags (both even and odd ID tags) is around 39 s. In comparison, if the queue discipline is FIFO from the beginning to the end (i.e. 100% on the horizontal axis), both tag types have the same average waiting times of around 38 s. Fig. 14 also shows the values of the same measures observed in the real world experiment (where the queue discipline was changed from FIFO to PRI after 47% of the service instances were completed). As shown in this figure, the observed average waiting time for even ID tags (i.e. that have higher priority) is around 10 s and the same value for odd ID tags is around 81 se, and the observed average waiting time for all tags (both even and odd ID tags) is around 35.5 s.

7. Discussion and conclusions

Planning, control, and monitoring of construction operations in general, and heavy construction and infrastructure projects in particular, are facilitated using simulation modeling. Queuing systems are typically abstracted and represented as part of the overall project model in most simulation systems. However, in the absence of precise data with high spatial and temporal accuracy, realistic modeling of arrival and service processes in queuing systems simulation is not a trivial task. To alleviate this problem, data collection and knowledge extraction with regard to the interaction of entities in a queuing system were investigated in this research. In particular, this paper discussed major queue properties, and described the designed algorithms for knowledge discovery in queuing systems. The validity and robustness of the developed methodology were assessed using a series of experiments and through collecting and mining empirical data.

The designed data collection and knowledge-discovery methodology was successful in finding the interarrival and service times, and more importantly the underlying discipline of queuing systems. A key observation was that despite the convenience of modeling interarrival and service times using the Exponential probability distribution, this distribution may not necessarily be the true representative for certain operational scenarios. In fact, the empirical analysis of results obtained from the experiments conducted in this research revealed that the Exponential distribution was in most cases the worst fit probability distribution. Although this is a legitimate finding only in the context of this study and should not be generalized to all queues, it suggests that the assumption of Exponential distribution for interarrival of entities to a queue may not be always correct. Furthermore, the designed queue discipline discovery and data mining algorithms were used in several experiments to analyze queue data and detect the predominant queue discipline, as well as any sudden changes in queue discipline during the course of the operations. In particular, three initial experiments were conducted to assess the queue properties discovery algorithms. Results indicated that these algorithms were able to successfully detect queue properties for use inside a DES model. Furthermore, in a separate experiment in which queue discipline was switched at an arbitrary point in time during the operation of the queuing system, the effect of discovering the exact point at which the queue discipline was changed on the results of the simulation was evaluated. It was found that incorporating knowledge describing a real queuing system into a data-driven simulation model (as opposed to using static input data) can result in

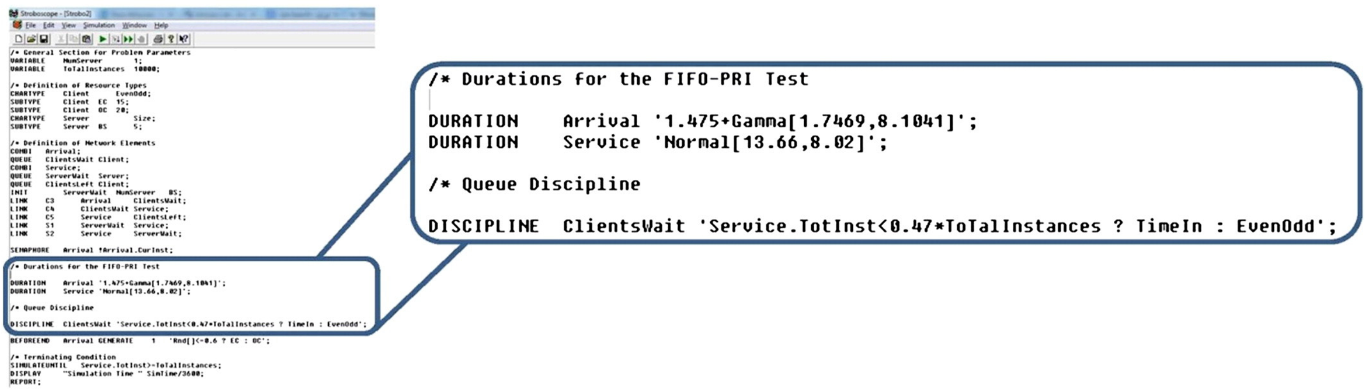

Fig. 13. STROBOSCOPE simulation script and the implementation of the discovered queue property change.

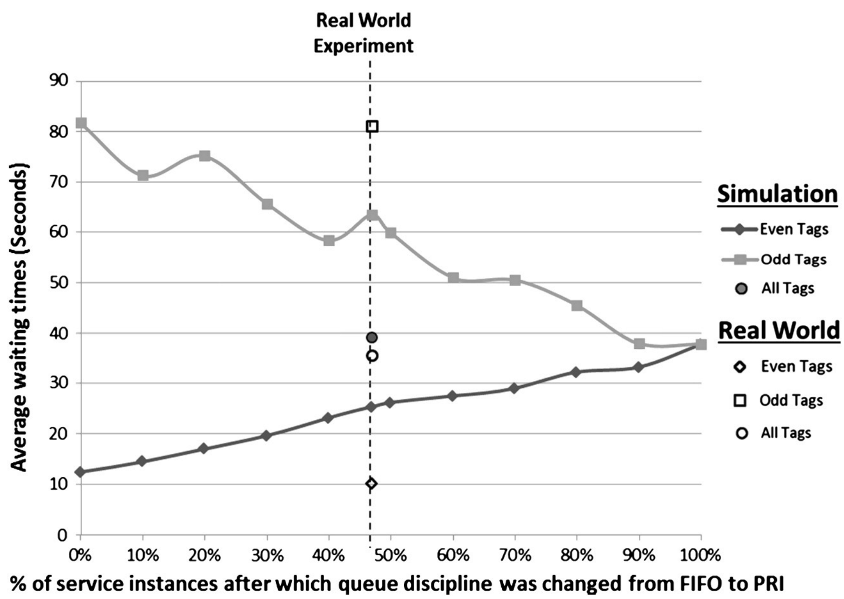

Fig. 14. Comparison of average waiting times of even and odd ID tags.

simulation outputs that better resemble the observations taken from the real system.

It was concluded from the validation results that relying solely on expert judgments, secondary (historical) data from past projects, and purely mathematical theories without considering the nature and unique characteristics of the current project may result in misrepresentation of the real system in a construction simulation model. This can adversely affect the reliability of the simulation output and make the results unacceptable for decision-making.

The main contribution of this research to the body of knowledge and practice is that it enables the extraction of queue properties in dynamic constantly-changing settings such as construction and infrastructure field activities. This is of practical value because queue modeling is one of the most important applications of simulation in general and DES modeling in particular. By providing factual data and extracting more reliable input knowledge such as queue properties for a simulation model, the output of the model is more representative and reliable. The authors are currently working on more complex examples of client–server interactions including multiple serving channels and/or phases, as well as cases for which more than one server processes a client simultaneously.

Acknowledgments

The authors would like to acknowledge the support of the UCF Office of Research and Commercialization (ORC) and the Wharton-Smith Construction Group. The authors would also like to thank Ms. Arezoo Shirazi, graduate student, and Ms. Brittany Cogger, undergraduate student, both members of the Decision Support, Information Modeling, and Automation Laboratory (DESIMAL) at UCF for their participation and assistance in conducting some of the experiments presented in this paper. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors or the individuals named above.

References

[1] R.B. Cooper, Introduction to queueing theory, North Holland, New York, 1981.
[2] J. Medhi, Stochastic models in queueing theory, Access Online via Elsevier, 2002.
[3] S. AbouRizk, Role of simulation in construction engineering and management, J. Constr. Eng. Manag. 136 (10) (2010) 1140–1153.
[4] J.C. Martinez, Methodology for conducting discrete-event simulation studies in construction engineering and management, J. Constr. Eng. Manag. 136 (1) (2009) 3–16.

[5] M. Lu, Simplified discrete-event simulation approach for construction simulation, J. Constr. Eng. Manag. 129 (5) (2003) 537–546.
[6] M. Arashpour, R. Wakefield, N. Blismas, E. Lee, Analysis of disruptions caused by construction field rework on productivity in residential projects, J. Constr. Eng. Manag. 140 (2) (2013) 04013053.
[7] I. Brodetskaia, R. Sacks, A. Shapira, Stabilizing production flow of interior and finishing works with reentrant flow in building construction, J. Constr. Eng. Manag. 139 (6) (2012) 665–674.
[8] S. Nunnally, Managing construction equipment, Prentice Hall, New Jersey, 2nd ed., Prentice-Hall, Upper Saddle River, N.J., 2000.
[9] D.W. Halpin, L.S. Riggs, Planning and analysis of construction operations, Wiley, New York, 1992.
[10] F. Farid, T.L. Koning, Simulation verifies queuing program for selecting loader-truck fleets, J. Constr. Eng. Manag. 120 (2) (1994) 386–404.
[11] A.M. Law, W.D. Kelton, Simulation modeling and analysis, McGraw Hill, Boston, MA, 2000.
[12] J.C. Martinez, Stroboscope: state and resource based simulation of construction processes, (thesis) University of Michigan, 1996.
[13] P.G. Ioannou, J.C. Martinez, Simulation of complex construction processes, in: D.M.J. Charnes, D. Brunner, J. Swain (Eds.), Proceedings of the 28th Conference on Winter Simulation, Institute of Electrical and Electronics Engineers, Inc., 1996, pp. 1321–1328.
[14] A.H. Behzadan, V.R. Kamat, Enabling smooth and scalable dynamic 3D visualization of discrete-event construction simulations in outdoor augmented reality, 2007 Simulation Conference (WSC), Washington, DC, 2007, pp. 2168–2176.
[15] S. AbouRizk, D. Halpin, Y. Mohamed, U. Hermann, Research in modeling and simulation for improving construction engineering operations, J. Constr. Eng. Manag. 137 (10) (2011) 843–852.
[16] S. Lee, A. Behzadan, A. Kandil, Y. Mohamed, Grand challenges in simulation for the architecture, engineering, construction and facility management industry, 2013.
[17] R. Akhavian, A.H. Behzadan, Knowledge-based simulation modeling of construction fleet operations using multimodal-process data mining, J. Constr. Eng. Manag. (2013) 04013021.
[18] R. Akhavian, A.H. Behzadan, An integrated data collection and analysis framework for remote monitoring and planning of construction operations, Adv. Eng. Inform. 26 (4) (2012) 749–761.
[19] R. Akhavian, A.H. Behzadan, Remote monitoring of dynamic construction processes using automated equipment tracking, Construction Research Congress 2012, ASCE, 2012, pp. 1360–1369.
[20] A. Hammad, C. Zhang, Towards real-time simulation of construction activities considering spatio-temporal resolution requirements for improving safety and productivity, in: R.R.C.S. Jain, J. Himmelspach, K.P. White, M. Fu (Eds.), Proceedings of the 2011 Winter Simulation Conference, Institute of Electrical and Electronics Engineers, Inc., 2011, pp. 3533–3544.
[21] L. Song, N.N. Eldin, Adaptive real-time tracking and simulation of heavy construction operations for look-ahead scheduling, Autom. Constr. 27 (2012) 32–39.
[22] P.G. Ioannou, Construction of a dam embankment with nonstationary queues, Proceedings of the 31st conference on winter simulation: simulation—a bridge to the future, Volume 2, ACM, 1999, pp. 921–928.
[23] L. Darren Graham, S.D. Smith, P. Dunlop, Lognormal distribution provides an optimum representation of the concrete delivery and placement process, J. Constr. Eng. Manag. 131 (2) (2005) 230–238.
[24] M. Oberguggenberger, Queueing models with fuzzy data in construction management, Analyzing uncertainty in civil engineering, Springer, 2005. 197–210.
[25] J. Banks, Discrete event system simulation, Pearson Education, New Jersey, 2005.
[26] J.F.C. Kingman, Poisson processes, Oxford University Press, USA, 1992.
[27] F.R. Jacobs, R.B. Chase, N.J. Aquilano, Operations and supply management, McGraw-Hill, New York, NY, 2009.
[28] D.G. Kendall, Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain, Ann. Math. Stat. (1953) 338–354.
[29] J. Banks, Handbook of simulation: principles, methodology, advances, applications, and practice, Wiley, New York, 1998.
[30] J.H. Willenbrock, Estimating costs of earthwork via simulation, J. Constr. Div. 98 (1) (1972) 49–60.
[31] S.M. AbouRizk, D.W. Halpin, J.R. Wilson, Fitting beta distributions based on sample data, J. Constr. Eng. Manag. 120 (2) (1994) 288–305.
[32] C. Maio, C. Schexnayder, K. Knutson, S. Weber, Probability distribution functions for construction simulation, J. Constr. Eng. Manag. 126 (4) (2000) 285–292.
[33] R.G. Miller, Priority queues, Ann. Math. Stat. 31 (1) (1960) 86–103.
[34] L. Takács, Priority queues, Oper. Res. 12 (1) (1964) 63–74.
[35] J.H. Steffen, J. Hotchkiss, Experimental test of airplane boarding methods, J. Air Transp. Manag. 18 (1) (2012) 64–67.
[36] M.H. DeGroot, M.J. Schervish, Probability and statistics, 3rd ed. Addison-Wesley, Boston, MA, 2002.