Deriving Generalised Stochastic Petri Net Performance Models from High-Precision Location Tracking Data

Nikolas Anastasiou
Department of Computing
Imperial College London
South Kensington Campus,
London SW7 2AZ
na405@doc.ic.ac.uk

Tzu-Ching Horng
Department of Computing
Imperial College London
South Kensington Campus,
London SW7 2AZ
th107@doc.ic.ac.uk

William Knottenbelt
Department of Computing
Imperial College London
South Kensington Campus,
London SW7 2AZ
wjk@doc.ic.ac.uk

ABSTRACT

Stochastic performance models have been widely used to analyse the performance and reliability of systems that involve the flow and processing of customers and/or resources with multiple service centres. However, the quality of performance analysis delivered by a model depends critically on the degree to which the model accurately represents the operations of the real system. This paper presents an automated technique which takes as input high-precision location tracking data – potentially collected from a real life system – and constructs a hierarchical Generalised Stochastic Petri Net performance model of the underlying system. We examine our method's effectiveness and accuracy through two case studies based on synthetic location tracking data.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modelling techniques

Keywords

Location Tracking, Performance Modelling, Data Mining, Generalised Stochastic Petri Nets

1. INTRODUCTION

We are surrounded by many complex physical systems that process customers and goods, for example hospitals, airports and car assembly lines. In such systems it is critical to understand the flow of customers and/or resources to ensure that the system can be tuned to meet its Quality of Service (QoS) requirements. For this reason, much time and effort has been invested in performance modelling and analysis. The traditional performance modelling and analysis pipeline consists of three stages: model construction, model validation and system analysis. The most fundamental stage of this pipeline is model construction as the accuracy of the model is crucial to ensure the validity of subsequent analysis. To build an accurate model usually requires a large amount of data. Current techniques, such as time and motion studies involve tedious manual tasks such as inspection of video footage, questionnaires and manual collection of timing data, all of which may be error-prone. This data-gathering process is not only time consuming but, importantly, often disrupts the system's natural flow.

Previous work on modelling patient flow in an Accident and Emergency department by S. Au-Yeung [2] exemplified the difficulties of manual model construction (particularly accurate parameterisation) and validation. While there was good agreement between *mean* response times emerging from both the model and the data, the *distributions* of response times were not well matched, and there was no straightforward way to identify the causes of discrepancies.

The recent development of real time location tracking systems (RTLSs) enables the automatic and unobtrusive collection of large amounts of high-precision location data in real time. These systems use a variety of technologies, such as RFID (Radio Frequency Identification), UWB (Ultra Wide Band) and Wi-Fi, and have been already deployed with the goal of enhancing performance efficiency and system safety, especially in the fields of supply chain management and healthcare (e.g. [1, 18, 11, 17, 14]).

As shown in Figure 1, the aim of this research is to design a methodology that automatically constructs a (stochastic) Petri Net Performance Model (PNPM) [3] of physical agent flow in a customer-processing system using location tracking data as input. Since this is an extremely broad and challenging problem for customer-processing systems in general, for the present we restrict ourselves to that class of systems having a single class of customer, single-server service semantics and random service discipline.

Our approach is based on the four-stage data processing pipeline shown in Figure 2, which takes as input raw location tracking data and outputs the PNPM in the portable PNML [4] format. The first stage of the pipeline performs some basic data filtering. The second stage infers the location of each service area in the system as well as its associated service radius. The third stage creates the initial structure of the PNPM with the required places and transitions. Samples of sojourn times in each service area and samples of travelling times between each pair of service areas are also extracted. The last part of this stage performs the calculation of the initial routing probabilities of the customer

flow. In the final stage we fit a hyper-Erlang distribution to each transition's assigned set of samples using the G-FIT tool [16] and refine the structure of the model accordingly. The output of the fourth stage – the PNML file – can be processed using a number of Petri net editors including PIPE2, an open-source platform independent Petri Net editor [6].

The remainder of this paper first presents previous literature related to our work. Then we examine the four stages of the developed processing pipeline in detail. Finally we present two case studies to examine the accuracy and effectiveness of our approach. These use synthetic location tracking data generated by a simulator (an extended version of JINQS [10]). We conclude the paper with a summary of the results and a discussion of future work.

2. RELATED WORK

A previous research endeavour in the field of automatic construction of Generalised Stochastic Petri Net (GSPN) [15] models has been made by Xue et *al.* [19]. It developed a methodology that automatically constructs GSPN models for flexible manufacturing systems (FMSs), which can respond to changes in the environment in real time. Their work included a software package, FMSPet, where an input language called FMSDL was used to describe the physical system being modelled. However, their methodology is application-specific and cannot be easily adapted to other scenarios. Our work aims to provide a generic tool which constructs accurate GSPN performance models representing a wide range of physical systems.

Our work is mostly closely related to earlier work by Horng et *al.* [12]. This work proposed a methodology for inferring simple Queueing Network performance models from high-precision location tracking data. The inferred Queueing Network model encapsulates both the structure of the Queueing Network, specified by customer-flow routing probabilities, and customer interarrival time and service time distributions for each server. Different from Horng et *al.*, this research adopts Petri nets, instead of queueing networks, as the modelling formalism. Furthermore, instead of assuming knowledge of service areas and service radii, we adopt a clustering algorithm to infer both the locations and radii of service areas (assumed to be circular). Our chosen clustering algorithm is DBSCAN [8], a density-based algorithm for discovering clusters in large spatial databases that contain noise. Clusters are defined as connected regions with high data density. If the data density of a region is less than a predefined threshold then these data are considered as noise. We apply DBSCAN on a set of 2D points extracted from filtered location traces, to identify the clusters that represent the service areas and approximate the locations of the service points by calculating their centroids.

This research also incorporates G-FIT, a technique for approximating general non-zero distributions by fitting mixtures of Erlang distributions (that is, hyper-Erlang distributions). Thümmler et *al.* extended the fitting procedure of [13], which is based on the expectation-maximisation (EM) algorithm. Similar to hyperexponential distributions, hyper-Erlang distributions are analytically tractable and can be easily used for numerical performance studies [13]; their more constrained form also allows for an efficient fitting algorithm [16]. However, compared to hyperexponential distributions the classes of distributions representable by hyper-Erlang distributions are extended significantly. Thümmler used several case studies based on both synthetic data and real traffic traces to demonstrate the effectiveness of G-FIT in approximating empirical data, compared to the other methods such as PhFit [5]. Other work in modelling time variables (e.g. users' cell residence time) in wireless networks and mobile computing systems has also demonstrated the generality of hyper-Erlang models [9].

3. INFERRING PNPMS FROM HIGH-PRECISION LOCATION TRACKING DATA

3.1 Data Processing Pipeline

This section provides details of the automated four-stage data processing pipeline shown in Figure 2. The input of the pipeline is raw location tracking data and its output is a GSPN model describing customer flow in the system. The output of each stage serves as the input of the next one.

3.2 Stage 1

The first stage of the processing pipeline is responsible for basic data filtering. Currently we support UWB-based location tracking data generated by a Ubisense RTLS, as well as synthetic location tracking data generated by an extended version of JINQS.

A typical location update reading from an RTLS is of the form (`tagName, type, time, x, y, z, stderr`). `tagName` is a unique identifier for each agent in the system and `type` indicates the category a tag belongs to. This category is application specific and it can be used as an indicator for further information regarding the particular tag. For example, if we have multiple customer classes, `type` can be used to denote the customer class of the particular tag. `time` is the timestamp of each location update and `x`, `y`, `z` are the location of the tag in a 3D Cartesian coordinate system. `stderr` is the location tracking system's estimate of the deviation between the position measurement and the real location.

This stage also removes unnecessary information such as the z-coordinate (we work in 2D) and erroneous location readings that are outside the possibilities of the experimental environment. The final structure of a location update is of the form (`tagName, type, time, x, y, stderr`).

3.3 Stage 2

This stage consists of a three-layer technique which infers the location and service radii of service areas in the system. Here we assume that customers stop or slow down while receiving service; we can thus identify the regions where the customer movement is relatively 'slow' as the likely service areas. The three layers are: velocity filtering, density filtering and the application of the DBSCAN clustering algorithm [8] (see Figure 3).

In the following subsections we use the notion of the *Eps-neighbourhood* of a point p [8], denoted by $N_{Eps}(p)$, which for a dataset D is defined as

$$N_{Eps}(p) = \{q \in D \mid dist(p, q) \leq Eps\}$$

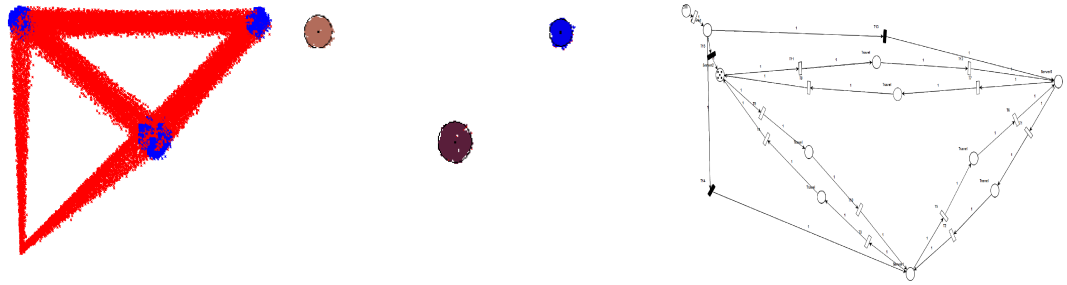


Figure 1: Our data processing pipeline takes as input raw location tracking data (left), as an intermediate stage infers the location of the stationary service areas and their associated service radius (middle) and outputs the inferred GSPN performance model (right).

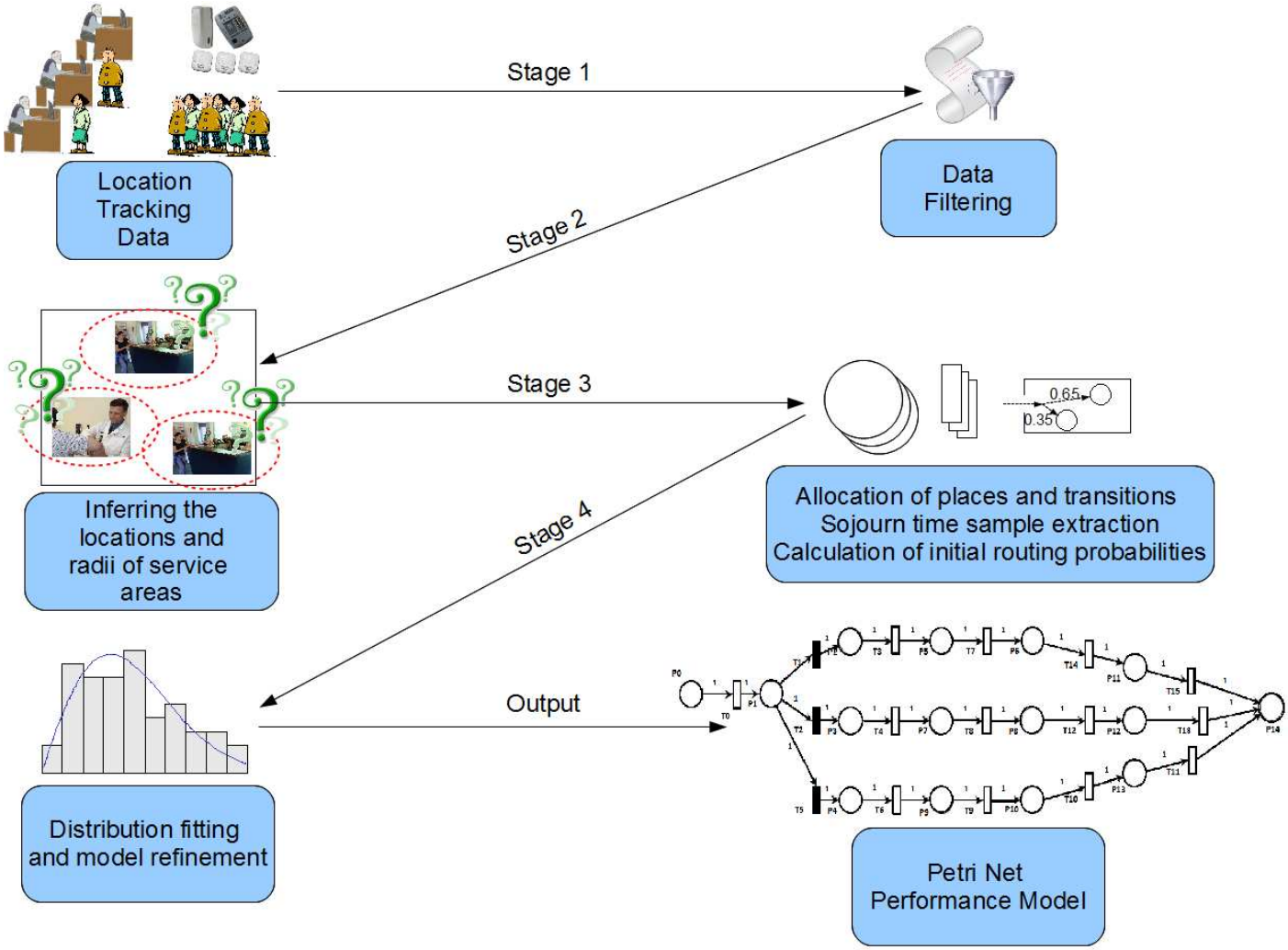


Figure 2: The four-stage data processing pipeline.

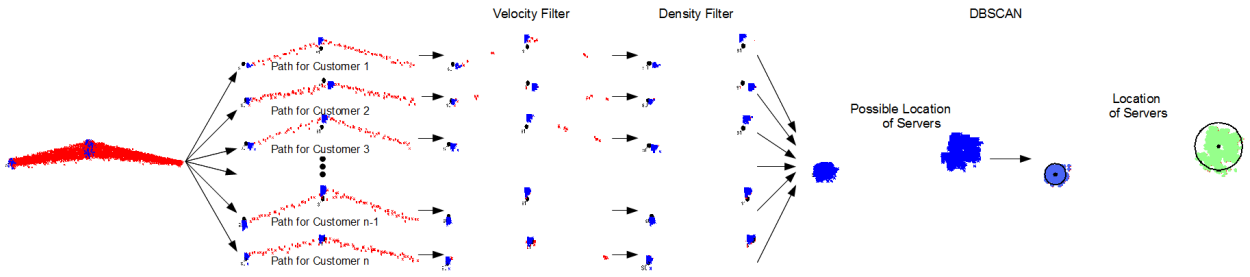


Figure 3: Stage two of the data processing pipeline.

That is, the set of points in D that lie in a circular area of radius Eps around the point p.

3.3.1 *Velocity filtering*

The location updates from Stage 1 are first separated into customer paths, one for each recorded customer. A customer path consists of the location traces of a particular customer's movements throughout its stay in the system. For each path, we calculate a raw velocity curve describing the average velocity of the customer for each time interval between consecutive readings. That is, if the distance between location readings at times t_i and t_{i+1} is d_{i+1}, the corresponding point on the raw velocity curve is $((t_i + t_{i+1})/2, d_{i+1}/(t_{i+1} - t_i))$. We then apply a moving average filter to produce a smoothed velocity curve (see Figure 4). The window size of the moving average is defined to be the 5% of the total number of readings in the particular customer's path.

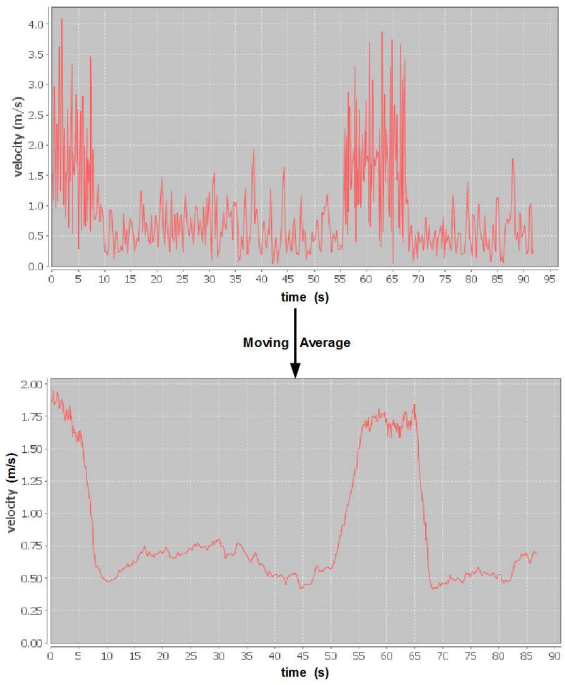


Figure 4: Application of the moving average filter on the raw velocity curve (above) to produce the smoothed velocity curve (below).

We then identify the local minima of each smoothed velocity curve. For each such minimum, we retrieve the corresponding location update data points, storing them as a time-sorted list, one for each customer.

3.3.2 *Density filtering*

The data points retrieved from the previous layer do not necessarily correspond to readings where the customer was in a service area since the customer may have simply paused en route between service areas. We apply a density filter to remove such points in regions of low data density.

To remove these points we use a threshold, $MinPts$, to specify the minimum number of points that should be present within the set $N_{Eps}(p)$, as well as a relatively small value for Eps. That is, for each point p in the velocity filtered data, if $|N_{Eps}(p)| \geq MinPts$, p remains in the list, else is discarded. In our implementation the value of $MinPts$ is chosen to be four, according to the suggestion of [8]. Eps is chosen to be 0.25 (metres) because the standard error of a typical RTLS is approximately this value. When this process is completed for every velocity filtered customer path, the remaining points from each path are merged (see Figure 3).

3.3.3 *DBSCAN clustering*

In the third layer the DBSCAN algorithm is applied to group points in the filtered dataset emerging from the previous layer into clusters, provided that they satisfy a density criterion. The challenge is to choose parameters for the density criterion that will best distinguish noise points from points that actually form the clusters corresponding to service areas. As before, this criterion is specified by two parameters, $MinPts'$ and Eps'.

We again choose $MinPts'$ to be four as our experience and that of others is that larger choices do not produce any significant difference in results while rapidly becoming computationally prohibitive [8]. The first step in finding a suitable value of Eps' is to compute the *4-dist* value for each point p. This is defined as the distance between p and its fourth-nearest neighbour [8]. We then compute and sort the *4-dist* values for all the points in the filtered dataset in descending order. A suitable value of Eps' is identified by finding the *4-dist* value of the "first valley" as shown in Figure 5. This value of Eps' best differentiates noise (points to the left of the valley) from points that potentially lie within service areas (points to the right of the valley). The value of Eps' can be manually selected or we can apply automated selection based on interpercentile distance.

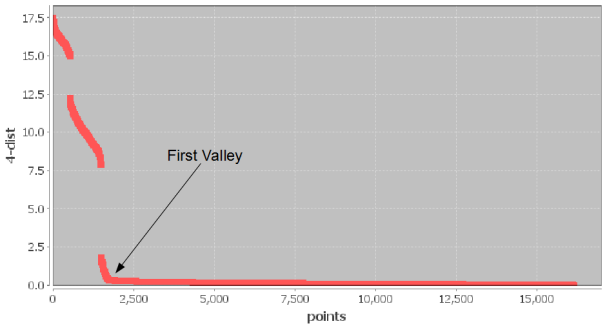


Figure 5: The sorted *4-dist* values for a sample filtered dataset.

The centroids of the formed clusters output by DBSCAN approximate the real locations of the corresponding service areas. The radius of each service area is conservatively approximated as 110% of the 95th percentile of the distance between each point in the cluster and the cluster's centroid.

3.4 Stage 3

Stage 3 constructs the basic structure of the derived PNPM. We first create places associated with the service areas inferred from the previous stage. The next step is to create places associated with customer movement between service areas, one for every pair of service areas between which customer movement was observed, and transitions connecting places representing service areas to those representing customer movement. We call these transitions service area service time transitions. Transitions are then created to connect the places associated with customer movement to places representing destination service areas. We call these transitions travelling time transitions. The resulting Petri net structure is illustrated in Figure 6.

For the moment, we do not parameterise the rates of the transitions; in fact in the next section we show how we replace each transition by a GSPN subnet that accurately reflects the distribution of the relevant time delays. In preparation for this, we compute – for each customer – samples of their sojourn times inside service areas (response time samples) broken down into waiting time and service time.

In order to estimate the service time a customer receives at a server's service area, we first estimate when the customer enters the service area (entry time) by taking the average of two timestamps called first appearance time and last disappearance time. Based on the customer's location traces, the first appearance time corresponds to the first timestamp when the customer is identified to be inside the server's service area; the last disappearance time is defined as the last timestamp when the customer is considered to be outside the service area. The customer's exit time is computed in a similar way, by taking the average of the timestamps of the two location updates that correspond to the last appearance and first disappearance [12]. We maintain two time-ordered lists at each service area to store the customers' entry and exit times. At each exit of a customer, we check first if the previous customer exit time is larger than this particular customer's entry time. In this case, the estimated service time is difference of the previous customer's exit time from the current customer's exit time. Otherwise, the server was idle upon the customer's arrival so the service time is simply the difference between the exit and entry times. We then compute travelling times by subtracting the customer's exit time at the upstream service area from the customer's entry time at the downstream service area along the customer's path.

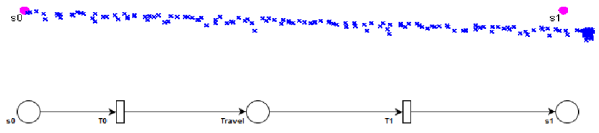


Figure 6: Trace of a customer movement from service area s0 to s1 (above) and its initial representation in a Petri net model (below).

One of the key challenges in this stage is to distinguish the cases where a customer simply passes through a service area without requesting service. To spot these cases, we compute the average velocity of each customer over a time window before it reaches a service area and compare it with the average velocity inside the service area. If the latter is less than the former we consider the customer having waited to be serviced by the server. A look-ahead action is also employed, in order to judge whether a real departure event occurred by checking if the departed customer returns to a server's service area within a short amount of time.

Finally, a simple counting mechanism is used to calculate the initial routing probabilities of the customer flow structure. These are represented as immediate transitions in the resulting PNPM.

3.5 Stage 4

The final stage aims to replace the service area service time transitions and travelling time transitions with GSPN subnets that accurately reflect the distributions of the corresponding service time and travelling time samples collected in Stage 3. As shown in Figure 7, the places and transitions in the GSPN subnets correspond to the phases of the hyper-Erlang distribution (HErD) which best fits the extracted timing samples.

The weights of the Erlang branches are represented by the immediate transitions. Each phase of an Erlang branch is represented by a timed transition in the GSPN.

We assume infinite server semantics for all timed transitions included in the subnet. In the representation of the service time transitions we have an additional complementary place to control the number of tokens (customers) that are allowed to be inside the subnet simultaneously. In this research, we assume service areas with single-server semantics and therefore the initial marking of this place is one. There is no such restriction on travelling time transitions.

To compute the candidate best-fit HErDs from our extracted timing samples, we make use of the G-FIT [16] tool. G-FIT's output includes the number of Erlang branches, their

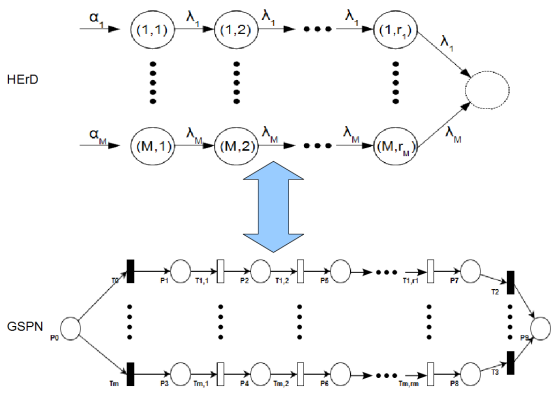


Figure 7: HErD to GSPN correspondence.

weights, the number of phases and the rate for each branch. We perform an exhaustive enumeration of possible HErDs up to a maximum number of total states. The best-fit HErD is chosen to be the one which achieves the lowest score under the Akaike Information Criterion (AIC) [7].

We have built in support for hierarchical visualisation of generated PNPM models into PIPE2, the open source Petri net editor. As shown in Figure 8, this allows for both uncluttered visualisation of the higher-level model structure (without full expansion of subnets) as well as on-demand detailed inspection of subnets.

4. CASE STUDIES

In this section we conduct two case studies in order to assess the applicability and accuracy of our approach. For the case studies we have generated location tracking data using an extended version of the simulator JINQS [10], the LocTrackJINQS. The synthetic data provide two advantages over real traces; they allow us to characterise the degree of accuracy of the inferred distributions and their parameters – as the exact model parameters and processes are known – and their generation is performed in a time efficient manner rather than engaging in long experimental procedures.

We focus on service area inference (second pipeline stage) and the extraction of the service area service time distributions (fourth stage). The experimental setup for each case study is depicted in Figure 9. The simulations take place in a virtual 25m × 25m environment with customer movements as illustrated. The customers are assumed to travel between servers in such a way that each journey has a velocity drawn from a normal distribution with mean 0.5 m/s and standard deviation 0.1 m/s for Case Study 1, and a velocity drawn from a normal distribution with mean 0.3 m/s and standard deviation 0.1 m/s for Case Study 2. The location update error is normally distributed with mean 0.15m and standard deviation 0.2m.

Each service area consists of a single customer processing server and a random customer service discipline. The service time for each server follows a different density function. Table 1 shows each server's actual location and service radius as well as its service time density for each case study.

		Server Location	Service Radius	Service Time Density
Case Study 1	Server 0	(2.0,2.0)	0.5	HErD(2,2;0.5,0.5;0.05,0.48)
	Server 1	(18.0,2.0)	0.8	Erlang(3,0.065)
	Server 2	(18.0,20.0)	0.35	Exp(0.1)
Case Study 2	Server 0	(5.0,5.0)	0.5	HErD(2,2,4;0.3,0.3,0.4;0.05,0.25,0.6)
	Server 1	(10.0,2.0)	0.75	Erlang(4,0.12)
	Server 2	(20.0,2.0)	0.3	Exp(0.18)
	Server 3	(10.0,8.0)	0.45	Normal(10.5,1.5)
	Server 4	(20.0,8.0)	1.5	Exp(0.04)

Table 1: The parameters for each server in the system, for each case study. The parameters of the HErDs represent the phase lengths, weights and rate for each branch respectively, separated by a semi-colon.

4.1 Results

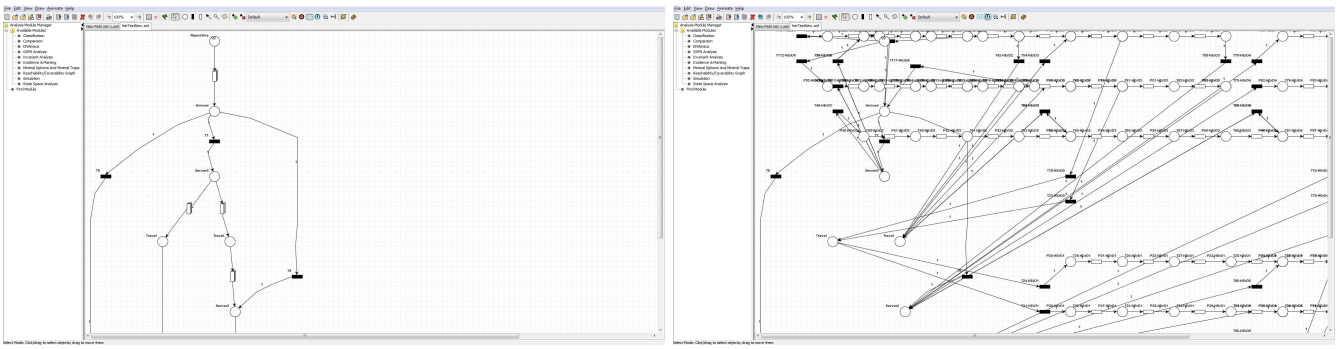
Figures 9(b) and 9(d) show the results of the second stage of our processing pipeline. Table 2 displays the estimated location and service radius for each service area as well as the error between the these and their real values (in terms of the distance between the real and inferred points).

From these results we can see that the inferred location matches the real location almost perfectly with a maximum error of 0.139 metres for the two case studies. The fitted radius for each server has larger error (max 0.289 metres).

For the purpose of evaluating the sample extraction and HErD fitting we compute the mean square error (MSE) between the theoretical and fitted distribution in Table 3. We also conduct a Kolmogorov–Smirnov test, examining the compatibility of the extracted service time samples for each service point with its best-fit HErD (see Table 4). The maximum number of states for the HErD to be fitted was set equal to ten, i.e. $N = 10$, for all cases when the coefficient of variation of the extracted sample is greater than 0.4 and twenty five, i.e. $N = 25$, when it is less.

Although the approximation of the actual service time density by the best-fit HErD is very good in all cases (see Figures 10 and 11), the parameters of the best-fit HErD do not match the parameters of the actual density in every case. For example, if we consider Server 0 in the first case study we see that the best fitted HErD has an additional branch with two phases and rate equal to 0.001. Its weight though is 0.006 and this suggests that the contribution from this branch is not of great significance. In some other cases, such as Server 2 in the second case study, the best-fit HErD differs significantly from the actual density. In this example one would expect to see a one branch HErD with one phase and rate close to 0.18.

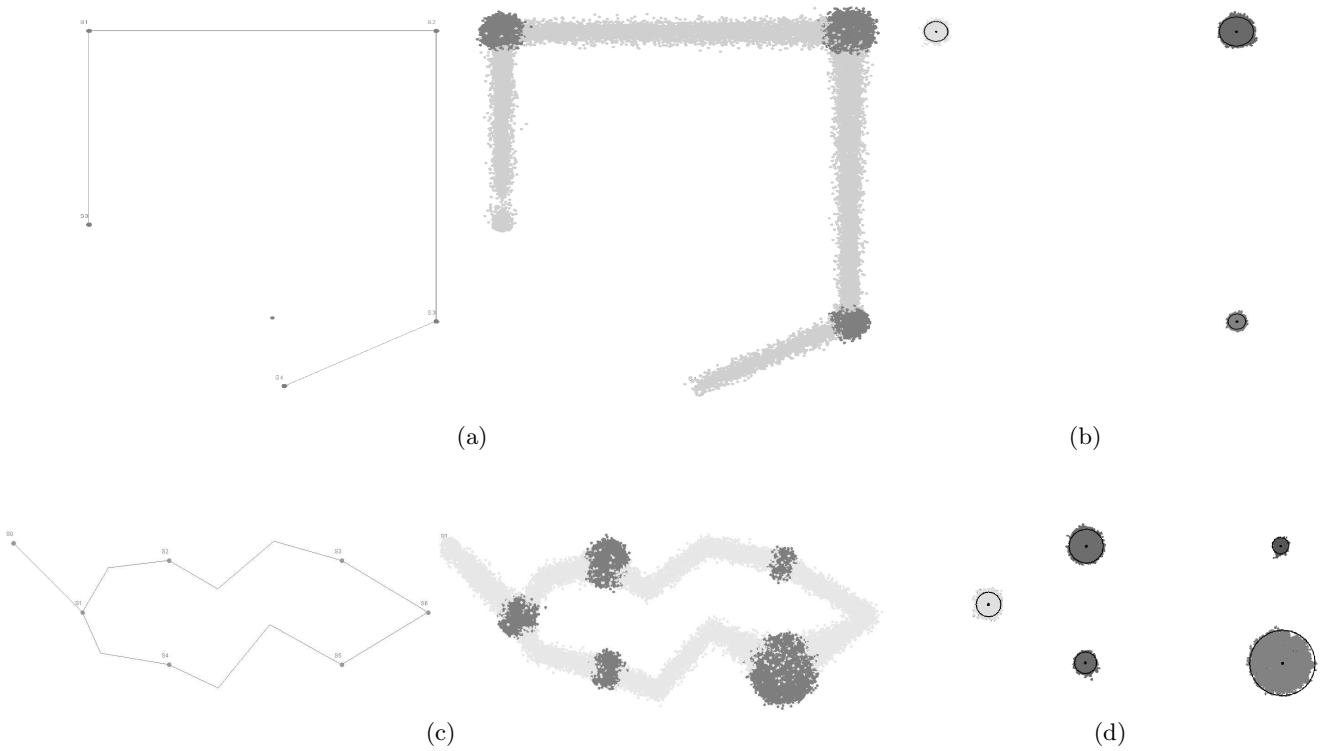
Figure 12 illustrates the inferred PNPMs. We observe that the structure of both models matches the abstract system structure of the models used to set up the simulation, i.e. for each server there exists a corresponding place and an asso-



(a) (b)

Figure 8: Support for hierarchical visualisation of generated PNPMs in PIPE2.

.



(a) (b)

(c) (d)

Figure 9: The experimental setup in terms of abstract system structure and generated location tracking data (a), (c) and clustering results (b), (d) of the first (above) and second (below) case studies. Light and dark traces indicate customer movement and stationarity respectively.

ciated subnet – in compact transition form – which models the server's service time. Between each pair of places representing servers we can also see intermediate travel places and their associated travelling time subnets.

5. CONCLUSION AND SUMMARY

This paper has presented a data processing pipeline for deriving Petri net performance models (in PNML format) from high-precision location tracking data collected from a class of simple customer-processing systems. The pipeline infers the location and radii of service areas in the system, extracts observed service time and travel time distributions and fits matching HErD distributions, resulting in a hierarchical GSPN model compatible with, amongst other tools, the PIPE2 open source Petri net editor.

The results of the two case studies indicate that the developed approach has the potential to infer the stochastic features of simple systems accurately, at least when synthetically-generated location tracking data is used.

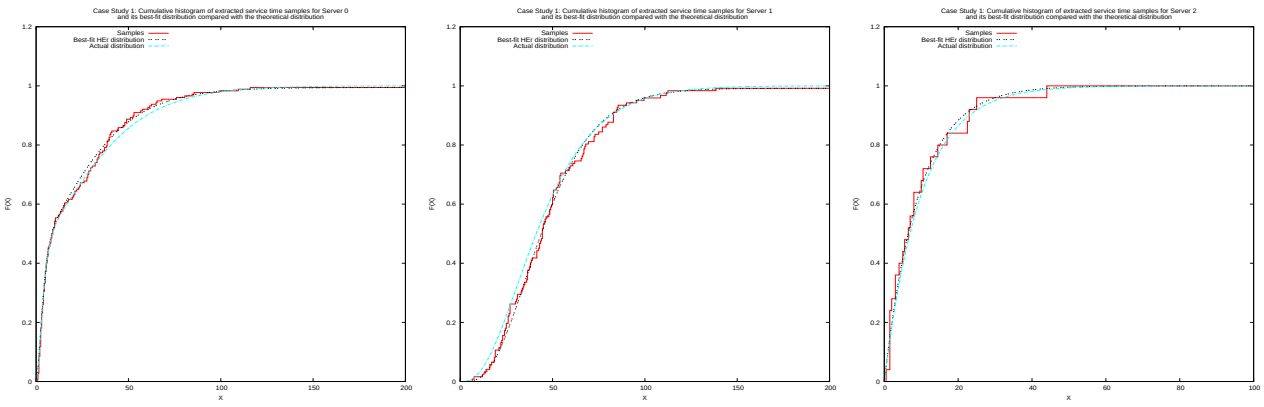
Currently our approach has several limitations which we intend to overcome in future work. For example, we have assumed a single class of customers, single-server service semantics, a single class of customers and a random service

		Server Location			Service Radius		
		Real	Inferred	Error	Real	Inferred	Absolute Error
Case Study 1	Server 0	(2.0,2.0)	(1.983,1.965)	0.039	0.5	0.723	0.223
	Server 1	(18.0,2.0)	(18.037,2.002)	0.037	0.8	1.007	0.207
	Server 2	(18.0,20.0)	(17.972,20.135)	0.139	0.35	0.513	0.153
Case Study 2	Server 0	(5.0,5.0)	(5.021,5.019)	0.028	0.5	0.694	0.194
	Server 1	(10.0,2.0)	(10.022,2.046)	0.051	0.75	0.976	0.226
	Server 2	(20.0,2.0)	(19.976,1.983)	0.029	0.3	0.589	0.289
	Server 3	(10.0,8.0)	(9.987,7.999)	0.013	0.45	0.662	0.212
	Server 4	(20.0,8.0)	(19.977,7.915)	0.088	1.5	1.678	0.178

Table 2: The inferred location and service radius for each server in the system accompanied with the absolute error, for each case study.

		Service Time Density	Fitted HErD Parameters			Mean Square Error (MSE)
			Phase Lengths	Rate (3 d.p.)	Weights (3 d.p.)	
Case Study 1	Server 0	HErD(2,2; 0.5,0.5; 0.05,0.48)	2,2,2	0.001,0.056,0.493	0.006,0.497,0.497	7.36E-5
	Server 1	Erlang(3,0.065)	4,4	0.001,0.084	0.008,0.992	2.64E-4
	Server 2	Exp(0.1)	1	0.107	1.0	2.44E-5
Case Study 2	Server 0	HErD(2,2,4;0.3,0.3,0.4;0.05,0.25,0.6)	2,4	0.061,0.567	0.335,0.665	5.41E-5
	Server 1	Erlang(4,0.12)	4	0.115	1.0	8.19E-5
	Server 2	Exp(0.18)	1,2,5	0.0002,0.273,3.120	0.013,0.790,0.197	2.58E-4
	Server 3	Normal(10.5,1.5)	25	2.278	1.0	1.15E-4
	Server 4	Exp(0.04)	1	0.042	1.0	3.00E-5

Table 3: The HErD parameters fitted by G-FIT for each server's service time density with the mean square error between the theoretical and fitted cumulative distribution function, for each case study. The parameters of the HErDs represent the phase lengths, weights and rate for each branch respectively, separated by a semi-colon.



(a) (b) (c)

Figure 10: Case Study 1: Graphs 10(a), 10(b) and 10(c) show the cumulative histogram of the extracted service time samples and its best-fit hyper-erlang distribution compared with the theoretical distribution for Server 0, Server 1 and Server 2 respectively.

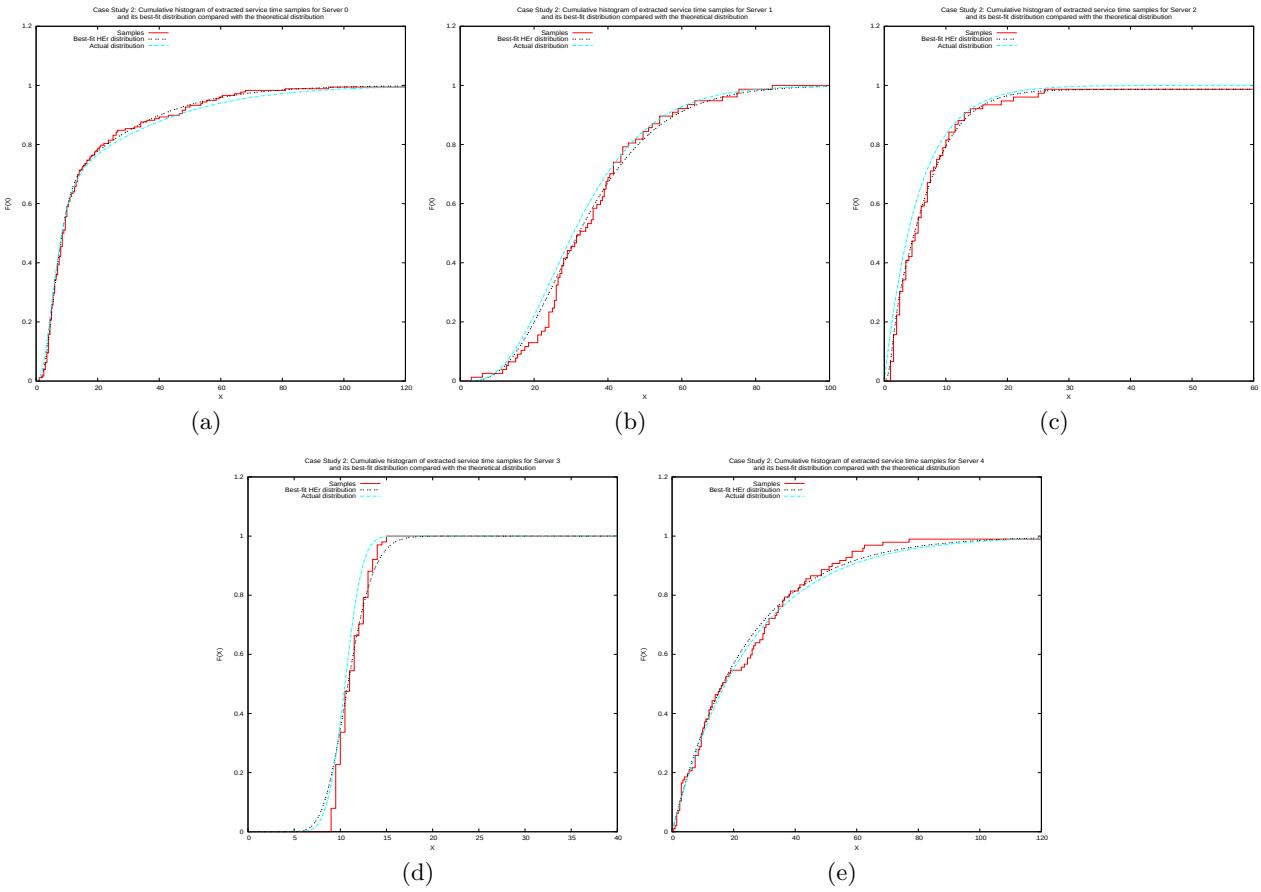


Figure 11: Case Study 2: Graphs 11(a), 11(b), 11(c), 11(d) and 11(e) show the cumulative histogram of the extracted service time samples and its best-fit hyper-erlang distribution compared with the theoretical distribution for Server 0, Server 1, Server 2, Server 3 and Server 4 respectively.

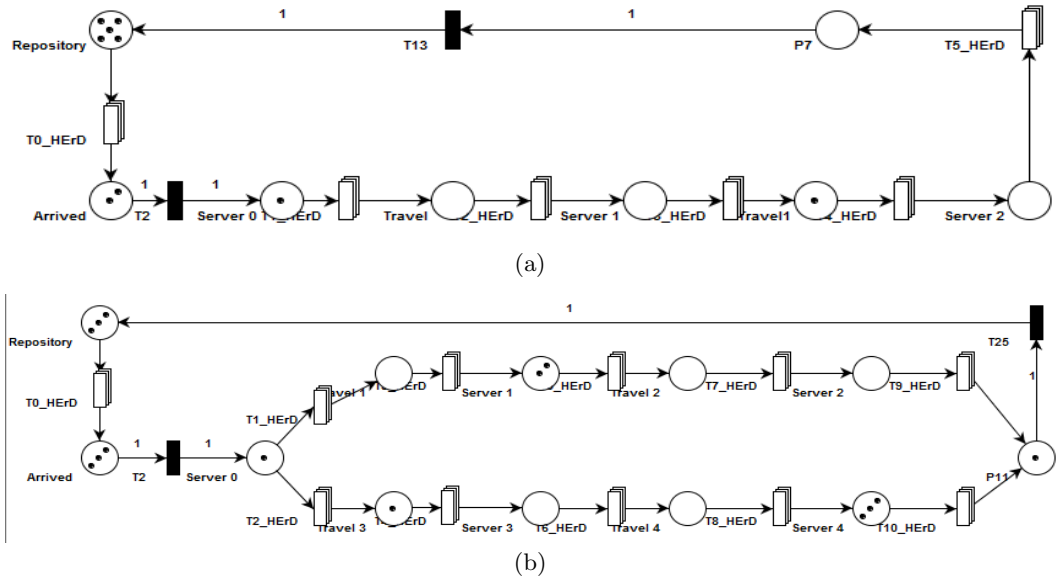


Figure 12: Visualization of the inferred GSPN performance models for case study 1 (above) and case study 2 (below).

		Case Study 1		Case Study 2	
Server 0	Test Statistic	0.0243		0.0261	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.0908	0.1008	0.0905	0.1005
	Compatible ?	Yes	Yes	Yes	Yes
Server 1	Test Statistic	0.0640		0.0212	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.1089	0.1209	0.1364	0.1515
	Compatible ?	Yes	Yes	Yes	Yes
Server 2	Test Statistic	4.2301E-5		0.0751	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.2330	0.2589	0.1372	0.1524
	Compatible ?	Yes	Yes	Yes	Yes
Server 3	Test Statistic	N/A		0.1129	
	α	N/A		0.1	0.05
	Critical Values	N/A		0.1195	0.1327
	Compatible ?	N/A		Yes	Yes
Server 4	Test Statistic	N/A		6.1348E-7	
	α	N/A		0.1	0.05
	Critical Values	N/A		0.1219	1353
	Compatible ?	N/A		Yes	Yes

Table 4: Kolmogorov-Smirnov test at significance levels 0.1 and 0.05 applied to the extracted service time samples for each service point from Case Studies one and two. The null hypothesis is that each extracted sample belongs to the corresponding best fitted HErD.

discipline. Key to supporting more general features, such as multiple customer classes and prioritised service disciplines, will be the use of Coloured Generalised Stochastic Petri nets (CGSPNs). Coloured tokens can also be used to control the routing of customers as they pass through various stages of processing. Multiple-server semantics may be implemented by increasing the number of tokens on the complementary place that controls admission to the service area, as well as appropriate adjustments to processing rates in the service time subnet.

Ultimately, we would like to examine the applicability of our pipeline in the context of more complicated scenarios using real location tracking data. While widespread adoption of our technique may currently be limited by a lack of such data, it is clear that this data will become increasingly available as ever-progressing wireless technology enables the realisation of the "internet of things", whereby vast numbers of everyday objects are networked and equipped with high-precision location tracking sensors.

6. REFERENCES

[1] R. Angeles. RFID Technologies: Supply-chain applications and implementation issues. *Information Systems Management*, (22):51–65, 2005.

[2] S. Au-Yeung. *Response Times in Healthcare Systems*. PhD thesis, Imperial College London, 2008.

[3] F. Bause and P. Kritzinger. *Stochastic Petri Nets*. Friedrich Vieweg & Sohn Verlag, 2002.

[4] J. Billington, S. Christensen, K. V. Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber. The Petri Net Markup Language: Concepts, Technology, and Tools. In *International Conference of Applications and Theory of Petri Nets Proceedings*, pages 1023–1024, 2003.

[5] A. Bobbio, A. Horváth, and M. Telek. PhFit: A General Phase-Type Fitting Tool. *International Conference on Dependable Systems and Networks*, page 543.

[6] P. Bonet, C. Llado, R. Puijaner, and W. Knottenbelt. PIPE v2.5: A Petri Net Tool for Performance Modelling. In *23rd Latin American Conference on Informatics (CLEI2007) Proccedings, San Jose, Costa Rica*, October 2007.

[7] H. Bozdogan. Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52:345–370, 1987.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference in Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon*, August 1996.

[9] Y. Fang. Hyper-Erlang distribution model and its application in wireless mobile networks. *Wireless Networks*, 7:211–219, 2001.

[10] T. Field. JINQS: An Extensible Library for Simulating Multiclass Queueing Networks V1.0. User Guide, 2006.

[11] T. Hansen, J. Bardram, and M. Soegaard. Moving out of the lab: Deploying pervasive technologies in a hospital. *IEEE Pervasive Computing*, 5(3):24 –31, 2006.

[12] T.-C. Horng, N. Dingle, A. Jackson, and W. Knottenbelt. Towards the automated inference of queueing network models from high-precision location tracking data. In *Proc. 23rd European Conference on Modelling and Simulation (ECMS 2009)*, pages 664–674, May 2009.

[13] R. E. A. Khayari, R. Sadre, and B. R. Haverkort. Fitting world-wide web request traces with the EM-algorithm. *Perform. Eval.*, 52(2-3):175–191, 2003.

[14] S. J. Kim, S. K. Yoo, H. O. Kim, H. S. Bae, J. J. Park, K. J. Seo, and B. C. Chang. Smart blood bag management system in a hospital environment. In *PWC '06: Proc. 11th International Conference on Personal Wireless Communications*, pages 506–517, 2006.

[15] M. Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.

[16] A. Thümmler, P. Buchholz, and M. Telek. A Novel Approach for Phase-Type Fitting with the EM Algorithm. *IEEE Transactions on Dependable and Secure Computing*, 3:245–258, 2005.

[17] M. Vankipuram, K. Kahol, T. Cohen, and V. L. Patel. Toward automated workflow analysis and visualization in clinical environments. *Journal of Biomedical Informatics*, In Press, Corrected Proof, 2010.

[18] F. Wu, F. Kuo, and L. Liu. The Application of RFID on Drug Safety of Inpatient Nursing Healthcare. In *ICEC '05: Proc. 7th International Conference on Electronic Commerce*, pages 85–92, August 2005.

[19] Y. Xue, R. Kieckhafer, and F. Choobineh. Automated construction of GSPN models for flexible manufacturing systems. *Computers in Industry*, 37:17–25, 1998.