A Tool for the Analysis of Hierarchical Service-Oriented Extended Open Fork/Join Queueing Networks

Markus Arns
Department of Computer Science
University of Dortmund
D-44221 Dortmund
markus.arms@uni-dortmund.de

ABSTRACT

In this paper we present a tool for the analysis of hierarchical service-oriented extended open fork/join queueing networks (EOFJQNs). The tool especially focusses on application areas that consider a service-oriented respectively process-oriented view of systems. Such application areas are for example business process management, logistics, supply chain management, production planing and control, and computer and communication systems. Since parallel process execution is a typical property of corresponding models we consider extended queueing networks that incorporate complex fork/join structures. For the analysis of EOFJQNs we apply a frequently used decomposition approach and we additionally approximate inter-arrival times and service times with phase-type distributions. Then the analysis of the isolated nodes is based on their underlying quasi-birth-and-death process. In this context the main problem is the analysis of complex fork/join structures. Therefore, at first we briefly reflect an approach for the analysis of rather simple fork/join nodes and afterwards apply a new aggregation techniques that allows us to reduce complex fork/join structures to simple fork/join nodes. We apply our tool to the analysis of a parallel production line.

Keywords

Fork/Join Queueing Networks, QBD-Analysis, Aggregation

1. INTRODUCTION

In this paper we present a tool for the steady-state analysis of hierarchical service-oriented extended open fork/join queueing networks (EOFJQNs). In contrast to alternative modelling and analysis frameworks for queueing networks [18, 12] our tool is especially intended for the analysis of service-oriented respectively process-oriented systems that additionally contain complex parallel processes. Such systems typically arise in application areas as for example business process management, logistics, supply chain management, production planning and control, and computer and communication systems. In queueing network terminology parallel processes are in general referred to as fork/join (sub-) networks.

For the analysis of EOFJQNs we apply a frequently used approximate decomposition approach [22, 31]. The main idea of this approach is to decompose a queueing network into isolated nodes that are analysed in isolation. The interrelation of the isolated subnets is realised due to the traffic flow. In our approach we approximate the traffic flow (inter-arrival time) and also the service requests to nodes with phase-type distributions and thereby map the internal behaviour of the isolated nodes to quasi-birth-and-death processes (QBDs). QBDs have the advantage that they can easily be analysed using matrix-geometric techniques [26]. Then the remaining problem that we have to solve is the QBD-based analysis of (complex) fork/join nodes.

Several authors have addressed to the analysis of fork/join nodes. Varki [28, 29] has extended the Mean Value Analysis for closed product–form queueing networks to the analysis of closed fork/join queueing networks. The fork/join nodes synchronise several single server queues with exponentially distributed service times. Baynat and Dallery [10, 9] apply Marie's product–form approximation [24] to the analysis of general closed fork/join queueing networks. In the context of manufacturing systems Krishnamurthy et al. [21, 20] applied special fork/join structures to the analysis of kanban control mechanisms. Furthermore, some results are known for the analysis of isolated fork/join nodes. In the special case of fork/join nodes with two parallel single servers with exponentially distributed inter–arrival and service times Flatto and Hahn [13] presented the generating function of the common queue length distribution. If additionally the service times are identically exponentially distributed Nelson and Tantawi [25] found exact results for the mean response times. In more general situations several approximations and bounds are known especially for the mean response times [6, 7, 30, 19, 5, 8, 23].

The fork/join nets that we consider are more flexible in the sense that they consist of an arbitrary but fixed number of complex parallel processes. These complex parallel processes may themselves be described in terms of EOFJQNs. For the analysis of fork/join nodes we proceed in two steps. At first we only consider a simple type that consists of several fcfs single-server queues in parallel with heterogeneous

SMCtools '07, October 26, 2007, Nantes, France

phase-type distributed service times and a common phase-type distributed inter-arrival process. The isolated (approximative) analysis of this simple fork/join node was presented by Balsamo et al. [8]. To handle fork/join nodes that synchronise more complex parallel processes that are beyond fcfs single-server queues we in a first step analyse each of the parallel networks individually under the common inter-arrival process with respect to the mean and the variance of the response times. In the second step we substitute the parallel networks with special single-server aggregates that yield the same mean and variance of the response times and thus reduce the analysis of extended (complex) fork/join nodes to the analysis of simple fork/join nodes.

The outline of our paper is as follows: In section 2 we present the class of queueing networks that our tool is intended for. We especially refer to the hierarchical service-oriented view of our approach. In section 3 we describe our analysis approach and mainly refer to the analysis of fork/join nodes. For the analysis of complex fork/join nodes we furthermore present our new aggregation technique. In section 4 we apply our tool to an example of a parallel production line that yields very accurate results with respect to the mean response time. Finally, we summarise our paper in section 5.

2. THE CLASS OF EOFJQNS

In our implementation we follow a hierarchical service-oriented view of EOFJQNs. Therefore, EOFJQNs consist of three parts, i.e.

1. a description of available resources,
2. a definition of service respectively process pattern and
3. the process instantiation.

The first part defines the available resources in terms of a set of nodes. The nodes may be of basic node-types or of complex node-types and they offer special services that define interfaces to the environment respectively the way the resources can be used by customers respectively processes. Currently, basic node-types are either fcfs nodes with one or multiple servers and infinite queueing capacity or infinite server nodes. Both types offer the default service *request* that allows a customer or a process to use the node for a predefined time period. We assume that this time period is described in terms of special phase-type distributions (see tab. 3.1) with respect to the first and the second moment. We will refer to complex node-types later.

The second part of EOFJQNs defines one or multiple service pattern (process pattern). A process pattern describes a chronological ordering of service requests to the available resources. Thereby, a process pattern may sequentially request some services, it may branch into alternative (sub-) processes (Markovian routing) and it may fork into several parallel/concurrent (sub-) processes. The branch respectively the fork operation may branch respectively fork into an arbitrary but fixed number of alternative respectively parallel subprocesses with an arbitrary complexity. We later refer to what we mean with the term *complexity* in section 3 in more detail. As the only restriction we require matching pairs of branch and end-branch operations as well as matching pairs of fork and join operations. So far,

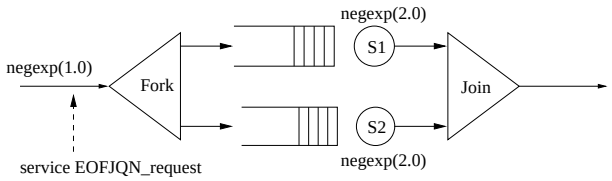

Figure 1: Example of an EOFJQN

an EOFJQN is nothing else but a compound component or complex node that offers one or more services (described in terms of process pattern) to its environment. We especially allow an EOFJQN to be part of the available resources of an environmental EOFJQN and thus get a hierarchical service-oriented architecture of EOFJQNs. We do not restrict the depth of hierarchies. In contrast to the basic node-types that expect the distribution of a time period upon service request the services of compound components may not have formal parameters in our current implementation. In the current version of our tool we only support single-class queueing networks which in practice means that each resource may only be used once in one process pattern.

The final part of an EOFJQN describes the instantiation of processes in terms of external arrivals. Again, we assume that external arrivals can be approximated by phase-type distributions according to tab. 3.1. In our current implementation the process instantiation part may only occur in the top-level hierarchy of an EOFJQN. To demonstrate our hierarchical service-oriented view of EOFJQNs we consider the example in fig. 1. The very simple example consists of a fork operation with two fcfs single-server nodes with infinite spatial capacity in parallel and a join operation. We refer to the semantics of fork/join nodes in section 3 in more detail. The external inter-arrival times as well as the service requests to the parallel servers are exponentially distributed with rate 1 respectively rate 2.

In the current implementation of our analyzer tool we support a command-line version that expects an XML-description of EOFJQN. Within a student project we are working on a graphical editor that allows us to graphically specify EOFJQNs and automatically transform them into the input language of our tool. The XML-description of the simple example in fig. 1 is displayed in fig. 2. The output of our tool is also an XML-description that adds the analysis results to the input description. We refer to the analysis results in section 3.

In the input specification the three parts of an EOFJQN are explicitly separated and enclosed by the `<availableResources>`, `<providedServices>` respectively `<extarrival` tags. From the above description the semantics of the XML-specification is easy to understand. We only mention that within the fork operation we separate the different parallel processes by enclosing them with a `<part>` tag. The same also holds for branch operations where we separate the different alternative branches with a `<part>` tag and additionally specify the branching probability.

The motivation for considering queueing networks as hierarchical service-oriented models arises from a portability point of view. In many application areas as for example software engineering, business process management, logistics networks, supply chain management, and production planning and control systems are modelled in a service-oriented or process-oriented fashion. Additionally, these models of-

```
<ExtendedQueueingNetwork name="EOFJQN_Example">
  <availableResources>
    <server name="S1" nserver="1" dis="FCFS"/>
    <server name="S2" nserver="1" dis="FCFS"/>
  </availableResources>
  <providedServices>
    <service name="EOFJQN_request">
      <fork>
        <part>
          <useService server="S1"
                      serviceName="request"
                      amount="negexp(2.0)"/>
        </part>
        <part>
          <useService server="S2"
                      serviceName="request"
                      amount="negexp(2.0)"/>
        </part>
      </fork>
    </service>
  </providedServices>
  <extarrival servicename="EOFJQN_request"
              interArrivalTimes="negexp(1.0)"/>
</ExtendedQueueingNetwork>
```

Figure 2: XML-description of an example EOFJQN

ten can be intuitively interpreted as queueing network models and thus it is natural to apply queueing network analysis techniques to theses models. To simplify model transformation and to meet the requirements of the aforementioned application areas we decided to reflect the service-oriented architecture in the structure of EOFJQNs.

3. ANALYSIS OF EOFJQNS

In this section we consider the analysis of EOFJQNs. We start with a description of the basic analysis technique and afterwards briefly present our approach for the analysis of extended (complex) fork/join nodes.

3.1 The basic analysis approach

For the steady-state analysis of EOFJQNs we apply the approximate decomposition approach by Kühn/Whitt [22, 31]. The main idea of this approach is to analyse the individual nodes of a queueing network in isolation and to realize the interconnection of the individual nodes with respect to their input/output behaviour (traffic flow). To make the analysis of EOFJQNs more tractable we furthermore approximate the input respectively output streams of nodes by special phase-type distributions with respect to the mean and the coefficient of variation. As proposed in [15] we select phase-type distributions depending on the coefficient of variation c according to table 3.1. Since we also assume that the service requests to the basic node-types are characterised by phase-type distributions a large set of basic node-types can be considered as quasi-birth-and-death (QBD) processes. QBDs can efficiently be analysed using matrix-geometric techniques (cf. [26]) and we can easily derive per node measures as the distribution of population

$c \geq 1$	hyper-exponential with 2 phases
$c = 1$	exponential
$c < 1$	hypo-exponential with $\lceil \frac{1}{c^2} \rceil$ phases

Table 1: Approximation with phase-type distributions

and response-time or the departure processes. Considering our basic node-types in section 2 only fcfs single-server nodes with infinite queueing capacity have a QBD representation. In case of multiple servers we additionally require the service requests to be exponentially distributed. (For non exponentially distributed service requests the node has no QBD representation.) Since also the infinite server has no QBD representation we approximate this basic node-type by a composition of several fcfs multiple-server nodes with exponentially distributed service requests (cf. [2]).

If we consider the traffic flow the sequential execution of service requests to different basic nodes means that we have to take the output process of a node and pass it to the input process of a subsequent node. To be more precise we approximate the output process by an adequate phase-type distribution as mentioned before and forward this approximation to the input process of a subsequent node. At this point several authors [16, 27] apply special Markovian Arrival Processes (MAPs) instead of phase-type distributions. MAPs have the advantage that they not only consider moments of departure processes but also incorporate correlations. In our implementation we also compute the output MAPs of the different node-types by truncation as proposed in [14, 17] and we can configure our tool such that it takes the departure MAP of a node as input MAP for a subsequent node. But from a practical point of view thereby the QBD representation of subsequent nodes grows dramatically in size and makes their analysis inefficient respectively impossible. Since on the other hand no appropriate general MAP fitting techniques are known to reduce the complexity of the departure MAPs we currently fit the departure processes by phase-type distributions. Considering the branch and end-branch operations we need appropriate splitting and superposition operations for phase-type distribution which is rather simple (see [15]). Thus, the major problem is the analysis of the complex fork/join processes. We will refer to this problem in the next subsection but prior we briefly refer to the performance measures that our tool computes.

Our tool computes per node measures as the first and second moment of the population, the response times and the inter-departure times in case of basic node-types and the first and second moment of the response times and the inter-departures times in case of compound node-types. Furthermore, we compute the first and the second moment of the response times and the inter-departure times per process. For this purpose we collect and accumulate the corresponding measures of the visited nodes. Of course, since the decomposition approach inherently neglects correlations among different nodes and due to our approximation of the traffic flow with phase-type distributions all the aforementioned measures can only be approximations.

3.2 Analysis of fork/join nodes

The analysis of fork/join nodes in queueing networks is

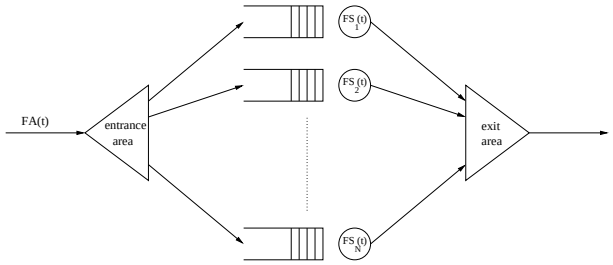

Figure 3: A simple fork/join model

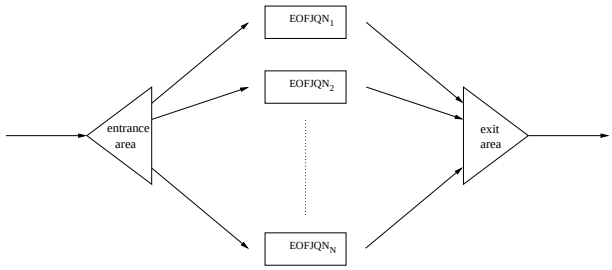

Figure 4: An extended fork/join model

a rather difficult task. Therefore, at first we consider the model of a simple fork/join node that is displayed in fig. 3. The model consists of an entrance area, N fcfs-single server nodes in parallel and an exit area. Upon arrival of a new customer or task at the entrance area the fork/join node immediately generates exactly N subtasks and passes them to the N parallel fcfs single-server nodes (fork operation). At the servers the subtasks eventually have to wait for service completions of previously arrived tasks. After the subtasks have finished service they immediately move to the exit area and wait for their siblings (join operation). If all subtasks have arrived to the exit area the corresponding task leaves the fork/join node without loss of time. At this point we remark that overtaking of subtasks is not possible due to the fcfs service discipline of the parallel servers. This means that at the exit area exactly that subtasks are joined that have been forked at the entrance area. Finally, we assume that the inter-arrival times of tasks at the entrance area as well as the service times of subtasks at the N parallel servers have phase-type distributions as mentioned above.

Unfortunately, this rather simple fork/join model does not have a QBD representation because the underlying Markov Chain is infinite in N directions. Thus, we cannot directly apply the above approach. To overcome this problem we consider an approximate model that was introduced by Balsamo et al. [8]. The idea of this approximate model is to introduce bounds $U_{ij} \geq 0, \quad i,j = \{1,\ldots,n\}$ that bound the number of subtask present at parallel server i to be at most U_{ij} greater than the number of subtasks present at server j for each pair of parallel servers (i,j). Let $z = (n_1,\ldots,n_N)$ be the state of this model with $n_i \geq 0$ subtasks at the i-th parallel server. Then the state space Z of this model is

$$Z = \{z = (n_1,\ldots,n_N) | n_i \geq 0, -U_{ji} \leq n_i - n_j \leq U_{ij}\} \quad (1)$$

Obviously, the arrival of a new task does not influence the bounds U_{uj} because each component of the state vector is simply incremented by 1. Thus, only the departure of a subtask at one of the parallel servers may violate one of these bounds. Therefore, consider servers i and j and let the number of subtask at server j be n_j and the number of subtasks at server i be $n_j + U_{ij}$. The departure of a subtask at server j now would violate bound U_{ij}. To prevent from that situation we simply block service at server j and we unblock server j upon the departure of a subtask at server i. Due to this blocking policy the mean response time of a task in this fork/join model is at least as long as the mean response time of the original model and by Little's law the same also hold for the mean population. In [8] Balsamo et al. give a formal proof of this aspect and hence they name this model *Upper Bound model.* To recognise the tridiagonal structure of the underlying QBD representation of the Upper Bound model we divide the state space Z into disjoint partitions $Z_k, k \geq 0$ such that

$$Z_k = \{z = (n_1,\ldots,n_N) \in Z | min_{i=1,\ldots,N}(n_i) = k\} \quad (2)$$

We easily see that $Z = \bigcup_{k=0}^{\infty} Z_k$ and obviously $z = (n_1,\ldots,n_N)$ is a state from partition Z_k if and only if $z' = (n_1+1,\ldots,n_N+1)$ is a state from Partition Z_{k+1}. That means that the partitions $Z_k, k \geq 0$ have equal cardinality. Furthermore, transitions only occur within the same partition or between directly neighboured partitions. Thus, the Upper Bound model has a QBD representation.

The QBD-based analysis of this Upper Bound model yields performance measures as the moments of the response time distribution of tasks, the distribution of the task population and the departure process and we can easily integrate the Upper Bound model into the above described decompositional analysis approach. We do not go into details of the QBD-based analysis here and we refer to [8] and [2]. Unfortunately, realistic models consist of much more complicated parallel processes that in general cannot be expressed in terms of several fcfs single-server nodes. We will refer to the analysis of such more complex extended fork/join nodes in the following subsection.

3.3 Analysis of extended fork/join nodes

In this section we consider extended fork/join models that differ from the simple fork/join models displayed in fig. 3 in the structure of the parallel processes. Beyond N parallel fcfs single-server nodes the extended fork/join model is composed of N parallel processes that are described in terms of EOFJQNs. The extended fork/join model is displayed in fig. 4. The behaviour of this model is similar to that of the simple model. Upon arrival of a new task at the entrance area the fork node immediately generates N subtasks that are passed to the N parallel processes $EOFJQN_1,\ldots,EOFJQN_N$. After the departure of a subtask at one of the $EOFJQN_i, i = 1,\ldots,N$ the subtask immediately moves to the exit area. In the extended fork/join node a departure occurs if N different subtasks (1 subtask from each of the $EOFJQN_i$) have arrived to the exit area. We explicitly remark that we cannot ensure that at the exit area exactly that subtasks are joined that have been forked at the entrance area. Consider for example an extended fork/join model with an infinite server node in one of the parallel $EOFJQN_i$.

In general the extended fork/join model has no QBD representation and also the state space of the QBD would be too large for practical application of the above mentioned anal-

ysis approach. Therefore, we have developed a new analysis approach for extended fork/join nodes. In this paper we only give a brief outline of our approach and we refer to [1] for more details. The main idea of our approach is to replace the complexity of the extended fork/join model displayed in fig. 4 to the complexity of the simple model displayed in fig. 3 and afterwards apply Balsamo's approximate analysis approach for the Upper Bound model. To reduce the complexity of the extended fork/join model in [1] we have introduced a new aggregation technique that allows us to replace each of the N parallel $EOFJQN_i, i = 1, \ldots, N$ with an *equivalent* fcfs single-server node. Thereby, *equivalent* means that the fcfs single-server aggregate yields exactly the same response time with respect to the mean and the variance as the corresponding $EOFJQN_i$ under the common inter-arrival times.

Again in the following we only give a brief overview of our aggregation technique and we refer to [1] for more details. We consider a fcfs single-server node with an unlimited waiting area and known phase-type distributed inter-arrival times. Furthermore, we assume that we have observed the mean E_R and the variance Var_R of the response times of tasks at this node. Then the question is whether we can find appropriate phase-type distributed service times such that the QBD-based analysis of this single-server node exactly yields the given moments of the response times under the given phase-type distributed inter-arrival times with rate λ_A and coefficient of variation c_A. To answer to this question first assume that the rate μ_S and the coefficient of variation c_S of the phase-type distributed service times were given. Then let $F_1(\lambda_A, c_A, \mu_S, c_S)$ respectively $F_2(\lambda_A, c_A, \mu_S, c_S)$ be the mean respectively the variance of the response times of the single-server node under given phase-type distributed inter-arrival and service times. We can compute F_1 and F_2 applying QBD-based analysis. In [1] we prove that for fixed values λ_A, c_A and c_S F_1 is continuous and strictly decreasing as a function in μ_S. We furthermore prove that for fixed values λ_A, c_A and μ_S F_2 is continuous and strictly increasing as a function in c_S. Returning to the above question if we knew the coefficient of variation c_S of the service times we could easily compute the rate of the service times as the (unique) root of

$$\tilde{F}_{c_S}(x) := F_1(\lambda_A, c_A, x, c_S) - E_R = 0. \quad (3)$$

For that purpose we could for example apply the *Regula falsi*. On the other hand if we knew the rate μ_S of the service times we could compute the coefficient of variation as the (unique) root of

$$\tilde{F}_{\mu_S}(y) := F_2(\lambda_A, c_A, \mu_S, y) - Var_R = 0. \quad (4)$$

From this observation in our aggregation approach we start with an initial value c_S (e.g. $c_S = 1$) and we iteratively (re-) compute $\mu_S = x$ respectively $c_S = y$ such that equation 3 respectively equation 4 holds. In [1] we present a necessary condition for the existence of an aggregate and we furthermore prove that the above procedure converges and thereby yields the rate and the coefficient of variation of the phase-type distributed service times of the aggregate.

Applying this results our algorithm for the analysis of extended fork/join nodes works as follows:

1. Analyse each of the N parallel $EOFJQN_1, \ldots, EOFJQN_N$ separately under the common inter-arrival times of the extended fork/join node.. Especially, compute the mean and the variance of the response times. For this step apply the decomposition approach described in subsection 3.1 to each of the complex parallel processes separately.
2. For each of the $EOFJQN_1, \ldots, EOFJQN_N$ determine corresponding fcfs single-server aggregates according to the previously described aggregation approach.
3. Replace each of the $EOFJQN_1, \ldots, EOFJQN_N$ with the corresponding aggregates and apply Balsamo's analysis approach for the Upper Bound model.

4. ANALYSIS OF A PRODUCTION-LINE

In this section we apply our tool to the analysis of a production line. As mention in section 2 our motivation for considering a hierarchical service-oriented view of queueing networks arises from the service-orientation respectively process-orientation of systems in several application area.

In logistics as well as in production planing and control systems are modelled in a process-oriented fashion in terms of so-called Process Chains. Process Chains mainly focus on the description of processes and therefore they define a series of activities that need to be performed in order to fulfil a process. The ordering of activities may be sequential, a process may branch into alternative subprocesses and it may fork into parallel subprocesses. Information on how activities are executed and which resources are available for that purpose often are only insufficiently specified. Thus, for the analysis of Process Chains this information have to be added and afterwards the Process Chains is (manually) transformed into a simulation model.

To overcome this laborious and error-prone procedure we have developed the ProC/B-tool[1] [4, 3] that integrates modelling and simulation of Process Chains. Therefore, we have defined a formalisation of Process Chains (the so-called ProC/B formalism) that not only describes processes but additionally gives information on available resources and the way the activities use these resources. Typically, resources are either time consuming (e.g. employees, machines etc.) or they are space consuming (e.g. inventory). The ProC/B-tool provides a graphical editor for modelling of (hierarchical) Process Chains respectively ProC/B models and it also allows a simulative investigation of ProC/B models. Therefore, it transforms a model into the input-language of the simulation tool HIT [11]. Recently, we also have implemented a translation of ProC/B models into the input-description of our analyser for EOFJQNs. This allows us to apply our approach to ProC/B models or at least to a subclass.

4.1 The ProC/B model

In this section we demonstrate our tool and therefore consider the ProC/B model of a production line. We do not go into the details of ProC/B but refer to [4, 3]. Our ProC/B model is displayed in fig. 5. The middle part of the model

[1]The ProC/B-tool was developed at the Department of Computer Science at the University of Dortmund in corporation with a collaborative Research Centre on "Modelling of Large Logistics Networks" (SFB 559, *www.sfb559.uni-dortmund.de*).

specifies the process and the bottom part describes the available resources. At the top of the model we see the model name and optional comments. We consider a company that produces special components for external customers. Orders from external customers arrive to the company in time-intervals that are approximated by a phase-type distribution with rate 1 and coefficient of variation 0.5. Upon the arrival of an order the company at first initiates some preparation (*prepare_oder*). This job is performed in the office (see the *Office* element in the resource part at the bottom of fig. 5). After the preparation step the company start the production of the order (see element *produce* in the middle part of fig. 5). The production is performed by the production unit.

The internal behaviour of the production unit is refined in the ProC/B model in fig. 6 on a deeper level of hierarchy. Upon arrival of an order (from the office) in the production unit the production unit immediately starts two parallel production processes that produce and inspect parts of type A respectively of type B. Obviously, this parallel production process has to be mapped to an extended fork/join node in the corresponding EOFJQN. We assume that the time that is needed for the production of type-A parts can be described in terms of a phase-type distribution with rate 2 and coefficient of variation 0.8. The inspection time be phase-type distributed with rate 6 and coefficient of variation 0.8. The information on type-B parts in fig. 6 have to be interpreted accordingly. After the parallel production and inspection of type-A and type-B parts in the final step both parts are assembled which takes a phase-type distributed time-period with rate 6 and coefficient of variation 0.8. To perform these five activities the production unit provides five teams of employees that serve orders in fcfs service discipline (see the bottom part of fig. 6).

4.2 Analysis of the ProC/B model

In the analysis of the above ProC/B model we are interested in the mean time-duration (response time) it takes from the arrival of an order to its fulfilment. Therefore, we have automatically transformed the above ProC/B model into the XML-input of our analyser tool. Besides the original parameterisation of the model we also have investigated how the mean response time changes if we increase the arrival rate of external orders. Thereby, we get an impression of the number of orders that the company can accept per time-unit while providing acceptable response times to its customers without changing the available resources. To get an impression of the accuracy of our approach we have compared the results to an simulative investigation of the model. The results of our experiments are displayed in table 4.2.

The table displays the arrival rate of external orders (λ) the mean response times calculated with our analyser tool (KW) the results from a simulative investigation (Sim) and finally the relative distance (Δ) of both results in percent. The simulation results lie in 90% confidence level of width smaller than 3%. Additionally, table 4.2 displays only that part of the (mean) response time that is spent in the parallel production process (extended fork/join node) of the production unit. The interpretation of the columns of table 4.2 is the same as for table 4.2.

Obviously, for this model our analysis approach yields very accurate results. We have applied our approach to several different logistics models with much more complex

λ	KW	Sim	Δ (%)
1.0	1.394	1.408	0.99
1.1	1.463	1.464	0.07
1.2	1.555	1.576	1.33
1.3	1.672	1.693	1.24
1.4	1.825	1.85	1.35
1.5	2.047	2.1	2.5
1.6	2.382	2.392	0.4
1.7	2.936	2.92	0.55
1.8	4.013	3.865	3.83

Table 2: mean response time

λ	KW	Sim	Δ (%)
1.0	1.052	1.062	0.94
1.1	1.12	1.115	0.45
1.2	1.2	1.222	1.8
1.3	1.323	1.338	1.12
1.4	1.474	1.491	1.14
1.5	1.693	1.738	2.59
1.6	2.045	2.025	0.99
1.7	2.577	2.546	1.24
1.8	3.652	3.52	3.75

Table 3: mean time spent in parallel production

fork/join subnets (cf. [1, 2]) and also observed very accurate results. One reason for the high accuracy of our approach in logistics networks is that those network typically are feed-forward nets and in general do not contain cycles. In case of cyclic nets the decomposition approach yields rather bad results especially with respect to response times due to the negligence of correlations among the nodes. Another reason for the high accuracy is the fact that we only have considered phase-type distributed external inter-arrival times. In several experiments on acyclic models with phase-type distributed external inter-arrival times we also have observed rather small correlations in the departure processes of inner nodes. Of course, this situation will dramatically change if we consider MAP-based external inter-arrival times. But as mentioned in section 3 we currently have no appropriate MAP-fitting techniques at hand that allow us to approximate the MAP-based internal traffic flow.

5. CONCLUSIONS

In this paper we have presented a tool[2] for the analysis of extended open fork/join queueing networks (EOFJQNs). The tool is based on an approximate decomposition approach and especially focusses on the analysis of queueing networks with complex fork/join structures. For the analysis of complex fork/join structures we have introduced an aggregation technique that allows us to reduce the complexity of fork/join structures to rather simple models that can easily be analyzed (approximatively) by a QBD-based approach.

To meet the requirements of service-oriented respectively

[2]Our tool is not available online yet.

process-oriented systems in a large range of application areas we follow a hierarchical service-oriented view of EOFJQNs. We have demonstrated the usefulness of this view in case of Process Chains that are an accepted formalism in logistics and production planing and control. The application of our tool to the analysis of a parallel production line yielded very accurate results with respect to the mean response time of the production line. Thereby, on the one hand we profit from the fact that Process Chains in general are acyclic networks and on the other hand from the description of the traffic flow in terms of special phase-type distributions. Within a student project we are currently working on a graphical editor that allows us to specify EOFJQNs and to automatically transform them into the input language of our tool.

6. REFERENCES

[1] M. Arns. A New Aggregation Technique for the Analysis of Extended Open Fork/Join Queueing Networks by Decomposition. *Proc. 13th GI/ITG Conf. Measuring, Modelling and Evaluation of Comp. and Communication Systems*, 2006.

[2] M. Arns. *Approximative Verfahren auf erweiterten Fork/Join-Warteschlangennetzen zur Analyse von Logistiknetzen*. PhD thesis, University of Dortmund, 2006.

[3] M. Arns, M. Eickhoff, M. Fischer, C. Tepper, and M. Völker. New Features in the ProC/B toolset. In *Tools of the 2003 Illinois International Multiconference on Measurement, Modelling, and Evaluation of Computer-Communication Systems*, Technical Report 781, Universität Dortmund, Fachbereich Informatik, 2003.

[4] M. Arns, M. Fischer, H. Tatlitürk, C. Tepper, and M. Völker. Modeling and Analysis Framework of Logistic Process Chains. In *Proc. of Joint Tool Session at PNPM/MMB/PAPM Conferences*, pages 56–61, Aachen, Germany, 2001.

[5] F. Bacelli. Two parallel queues created by arrivals with two demands: The M/G/2 symmetrical case. Technical Report 426, INRIA Rocquencourt.

[6] F. Bacelli, A. Makowski, and A. Shwartz. The fork-join queue and related systems with synchronization constraints. *Advances in Applied Probability*, 21:629–660, 1989.

[7] F. Bacelli, W. Massey, and D. Towsley. Acyclic Fork-Join Queueing Networks. *Journal of the ACM*, 36:615–642, 1989.

[8] S. Balsamo, L. Donatiello, and N. van Dijk. Bounded performance analysis of parallel processing systems. *IEEE Transactions on Parallel and Distributed Systems*, 1998.

[9] B. Baynat and Y. Dallery. A product–form approximation method for general closed queueing networks with several classes of customers. *Performance Evaluation*, 24:165–188, 1996.

[10] B. Baynat and Y. Dallery. An approximation method for general closed queueing networks with Fork/Join mechanism. *Journal of the Operational Research Society*, 51:198–208, 2000.

[11] H. Beilner, J. M"ater, and C. Wysocki. The Hierarchical Evaluation Tool HIT. *Short Papers and Tool Descriptions of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 1994.

[12] M. Bertoli, G. Casale, and G. Serazzi. Java Modelling Tools: an Open Source Suite for Queueing Network Modelling and Workload Analysis. *3rd Int. Conf. on the Quantitative Evaluation of Systems (QEST'06)*, 2006.

[13] L. Flatto and S. Hahn. Two Parallel Queues Created by Arrivals with Two Demands I. *SIAM J. Applied Math.*, 44:1041–1053, 1984.

[14] D. Green. *Departure Processes from MAP/PH!1 Queues*. PhD thesis, Departement of Applied Mathematics, University of Adelaide, 1999.

[15] B. Haverkort. Approximate Analysis of Networks of PH/PH/1/K Queues: Theory & Tool Suuport. In H. Beilner and F. Bause, editors, *Quantitative Evaluation of Computing and Communication Systems*, Lecture Notes in Computer Science 977, pages 239–253. Springer, 1995.

[16] A. Heindl and M. Telek. Output models of MAP/PH/1(/K) queues for an efficient network decomposition. *Performance Evaluation*, 49(1-4):321–339, 2002.

[17] A. Heindl, Q. Zhang, and E. Smirni. ETAQA Truncation Models for the MAP/MAP/1 Departure Process. *Proc. 1st Int. Conference on Quantitative Evaluation of Systems (QEST)*, 2004.

[18] H. Kaur, D. Manjunath, and S. Bose. The Queuing Network Analysis Tool (QNAT). *(th IEEE Int. Symp. on Modelling, Analysis, and Simulation of Com. and Telecommunication System (MASCOTS'00)*, 2000.

[19] C. Kim and A. Agrawala. Analysis of the fork-join queue. *IEEE Transactions on Computers*, 38:250–255, 1989.

[20] A. Krishnamurthy and R. Suri. Analytical Models for Pull-type Control Strategies in Manufacturing Systems. In *Proceedings of the Industrial Engineering Research Conference*, Houston, May 2004.

[21] A. Krishnamurthy, R. Suri, and M. Vernon. A New Approach for Analyzing Queueing Models of Material Control Strategies in Manufacturing Systems. In *Proc. 4th Int. Workshop on Queueing Networks with Finite Capacity (QNETs2000)*, West Yorkshire, U.K., July 2000.

[22] P. Kühn. Analysis of complex queueing networks by decomposition. *Congressbook, 8th ITC*, pages 236–1/8, 1976.

[23] J. Lui, R. Muntz, and D. Towsley. Computing Performance Bounds of Fork-Join Parallel Programs Under a Multiprocessing Environment. *IEEE Transaction on Parallel and Distributed Systems*, 9(3):295–311, 1998.

[24] R. Marie. An Approximate Analytical Method for General Queueing Networks. *IEEE Transactions on Software Engineering*, 5(5):530–538, 1979.

[25] R. Nelson and A. Tantawi. Approximate Analysis of Fork/Join Synchronization in Parallel Queues. *IEEE Trans. Computers*, 37(6):739–743, 1988.

[26] M. Neuts. *Matrix–Geometric Solutions in Stochatsic Models*. John Hopkins University Press, 1981.

[27] R. Sadre and B. Haverkort. Flows in Networks of

MAP/MAP/1 Queues. In *Proc. 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, pages 195–208, 2001.

[28] E. Varki. Response time analysis of closed fork-join networks. Technical Report, University of New Hampshire, 1998.

[29] E. Varki and S. Wang. A performance model of disk array storage systems. In *The Computer Measurement Group's 2000 International Conference*, Orlando, Florida, 2000.

[30] S. Varma and A. Makowski. Interpolation approximations for symmetric fork-join queues. *Performance Evaluation*, 20:245–265, 1994.

[31] W. Whitt. The Queueing Network Analyzer. *The Bell System Technical Journal*, 62(9):2779–2815, November 1983.

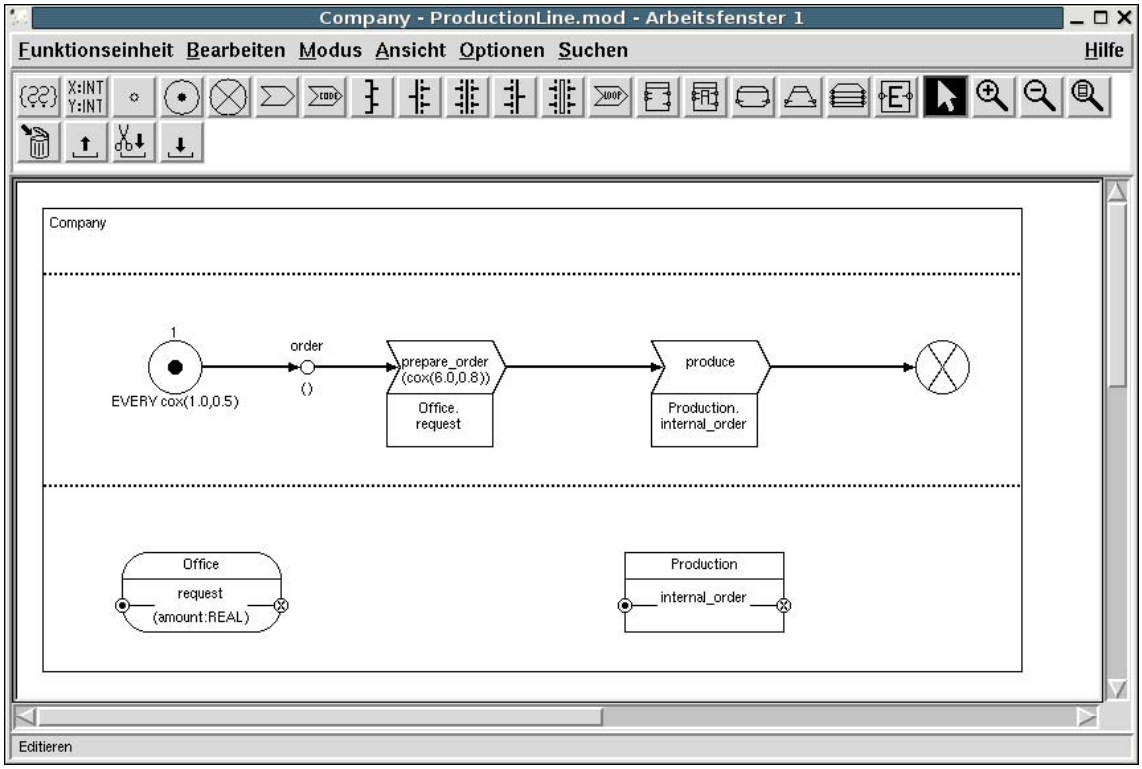

Figure 5: ProC/B model of the company

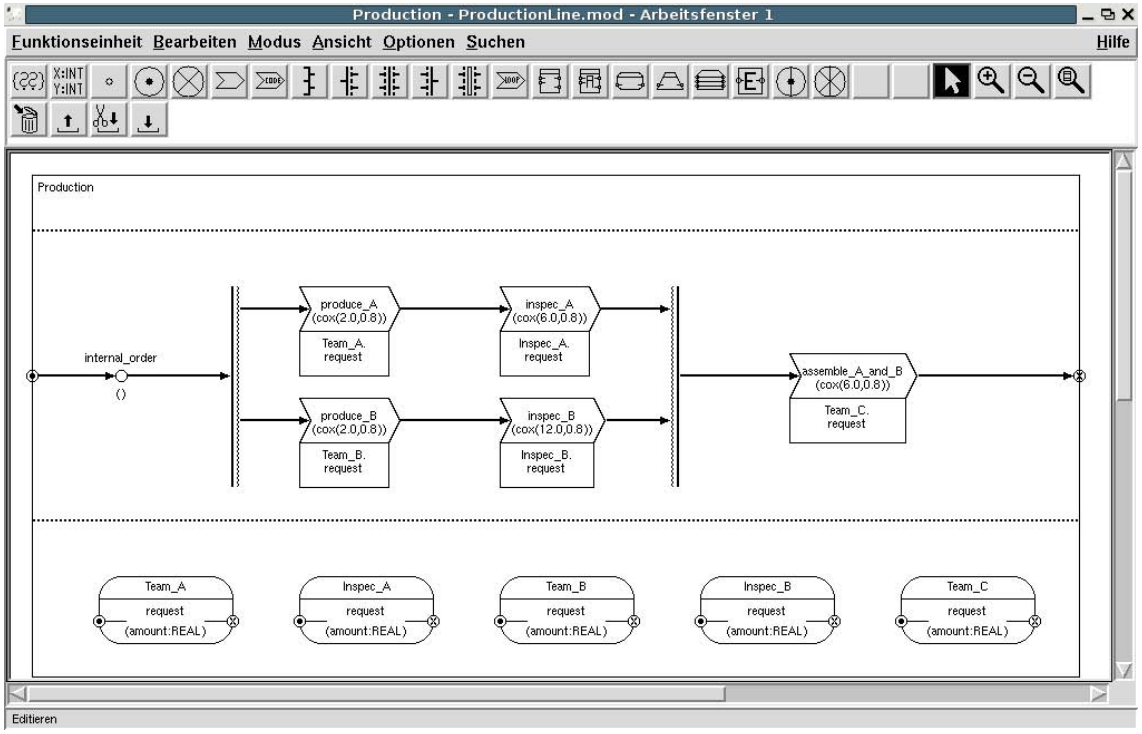

Figure 6: ProC/B model of the production unit