# Performance Assessment and Configuration of Enterprise-Wide Workflow Management Systems

## (Extended abstract)

**Michael Gillmann[1], Jeanine Weissenfels[1], Gerhard Weikum[1], Achim Kraiss[2]**

[1]University of the Saarland
Department of Computer Science
P.O.Box 15 11 50
D-66041 Saarbrücken
{gillmann,weissenfels,weikum}@cs.uni-sb.de
http://www-dbs.cs.uni-sb.de/

[2]Dresdner Bank AG
Organization and Information Technology
IT Research and Standards
Jürgen-Ponto-Platz 1
D-60301 Frankfurt a.M.
achim.kraiss@dresdner-bank.com
http://www.dresdner-bank.com/

## 1 Introduction

The main goal of workflow management systems (WFMS) is to support the efficient, largely automated execution of business processes. Large enterprises demand the reliable execution of a wide variety of workflow types. For some of these workflow types, the availability of the components of the underlying, often distributed WFMS is crucial; for other workflow types, high throughput and short response times are required. However, finding a configuration of the WFMS (e.g., with replicated components) that meets all requirements is a non-trivial problem. Moreover, it may be necessary to adapt the configuration over time due to changes of the workflow load, e.g., upon adding new workflow types. Therefore, it is not sufficient to find an appropriate initial configuration; it should rather be possible to reconfigure the WFMS dynamically. The first step towards a dynamic configuration tool is the analysis of the WFMS to predict the performance and the availability that would be achievable under a new configuration, and to determine the best configuration for the current workflow load.

In this paper, we present an analytic approach that considers the performance as well as the availability of the WFMS in its assessment of the quality of a given configuration of a distributed WFMS. The approach is based on well known stochastic methods [Nel95, STP96, Tij94] and shows the suitability of these models to a new application field. The presented combination of the methods allows an analytic assessment of WFMS eliminating the usual time- and cost-intensive trial-and-error practice for system configuration. Particularly, we are able to rank the performance and the availability of different configurations which use replicated workflow servers. Moreover, we can predict the performance degradation caused by transient failures of servers. These considerations lead to the notion of performability [STP96], a combination of performance and availability metrics. Likewise, we are able to calculate the necessary number of workflow server replications to meet the specified requirements for performance and availability. So, a crucial part of a configuration tool for distributed WFMS becomes analytically tractable and no longer depends on trial-and-error practices or the subjective intuition of the system administration staff.

Although the literature includes much work on scalable WFMS architectures, there are only few research projects that have looked into the quantitative assessment of WFMS configurations with regard to performance and availability [DKO+98]. [BD99] presents several types of distributed WFMS architectures and discusses the influence of different distribution methods on the network

This work was performed within the research project "Architecture, Configuration and Administration of Large Workflow Management Systems" funded by the German Science Foundation (DFG).

and workflow server load, based on simulations. In [BD97], the sustainable throughput of a distributed WFMS is increased by assigning subworkflows to appropriate workflow servers, based on online statistics about network partitions, network load, and expected communication costs. [SNS99] presents simple heuristics for the allocation of workflow type and workflow instance data onto servers. Work on WFMS availability has been presented in [HA98, KAG+96] that discuss how to efficiently increase the availability of process support systems by using standby mechanisms that allow a backup server to take over in the case of server failures.
The rest of the paper is organized as follows. In Section 2, we present our model of a distributed WFMS. In Sections 3 and 4, we develop a performance model and an availability model. In Section 5, we combine both models into the performability model that allows us to predict the influence of transient failures on the overall performance. Section 6 concludes the paper with a summary and an outlook on ongoing work.

## 2 System model of distributed WFMS

In this section, we describe a generic model of enterprise-wide distributed WFMS. Although the chosen system model is simple, it is powerful enough that we are able to capture the architecture models of most WFMS products and research prototypes in a reasonable way. Based on this model, we will introduce the central notions of the state and the configuration of a distributed WFMS. Finally, we present a model that stochastically describes the behavior of a single workflow instance.
Typically, distributed WFMS execute workflow instances in a partitioned and distributed manner, i.e., the workflow instance is partitioned into several subworkflows, which are executed in a distributed way on different workflow engines (e.g., with one workflow engine per partition/subworkflow type). These workflow engines typically run on several server machines distributed across an Intranet or even the Internet. Moreover, services that are "imported" from external companies can be integrated into the WFMS as subworkflows, and the WFMS of such a provider merely becomes another kind of workflow engine with a specific interface. The communication is handled by separate components, such as object request brokers (ORBs), other modules are responsible for workflow-specific functions like worklist management or monitoring, and the runtime state data of workflows is often stored in a DBMS. Finally, applications that are invoked from workflow activities may be spawned on dedicated application servers. All these components will be viewed as abstract servers of specific types in our system model.

### 2.1 Workflow server model

In workflows like the above examples, several cooperating components of a distributed WFMS are involved in the execution of a single workflow instance. We refer such to each component as a *server type*. For scalability and availability reasons, nearly all WFMS, both products and research prototypes, provide the replication of (performance-critical) server types within the system. Note that we have so far not said anything about the hardware (i.e., the server machines) that the workflow servers run on. It is possible (and often favorably or even unavoidable) to start workflow servers of different server types on the same server machine.
Figure 1 shows an example of the workflow server model. The dotted arcs indicate service requests between the several server types. But also within a single server type, workflow servers can request services from each other. For example, the execution of a subworkflow by a workflow engine that is not the same as the engine of the parent workflow is a service request, too. In our model, the communication services are only provided by some kind of *communication servers* such as object request brokers (ORBs) or TP monitors. The other server types do not communicate directly with each other but only via a communication server. In Figure 1, this is indicated by

the solid lines.

In our system model, every server type $s$ has a failure rate $\mu_s$ and a repair rate $\alpha_s$ (i.e., restart speed after a failure). For simplicity, we assume that the time between two successive failures of a server type as well as the time to repair one workflow server are exponentially distributed with the expected values $1/\mu_s$ and $1/\alpha_s$, respectively.

## 2.2 Configuration of a distributed WFMS

With the presented workflow server model at hand, we are now able to define the central notion of the system configuration of a distributed WFMS.

Because of failures and repairs of workflow servers, the number of available workflow servers of one server type varies over time. For a given point of time, we call the vector $(X_1, X_2, ..., X_N)$ of the numbers of currently available workflow servers of each workflow server type the current *system state* of the WFMS. The initial system state of the WFMS, i.e., the system state with all workflow servers available, is called the *system configuration* of the WFMS.

The goal of this work is to build a configuration tool, based on the above system model, that is able to derive the best system configuration for a given workflow load. The configuration tool should aim to optimize the ratio of performance and cost or availability and cost, respectively, or even the combination of both, the performability. We will discuss our approaches to these three kinds of goals in the following sections. As all such optimizations depend on the (probable) behavior of workflow instances, we first need to introduce an appropriate model for the control flow of a workflow instance.

## 2.3 Stochastic modeling of control flow

For predicting the expected load induced by the execution of a workflow instance, we have to be able to predict the control flow of workflow instances. As workflows include conditional branches and loops, the best we can do here is to describe the execution stochastically.

A suitable stochastic model for describing the control flow of a simple workflow instance without subworkflows is the model of continuous-time, first-order Markov chains (CTMC) [Nel95, STP96, Tij94]. A CTMC is a process that proceeds through a set of states in certain time periods. Its basic property is that the probability of entering the next state within some time only depends

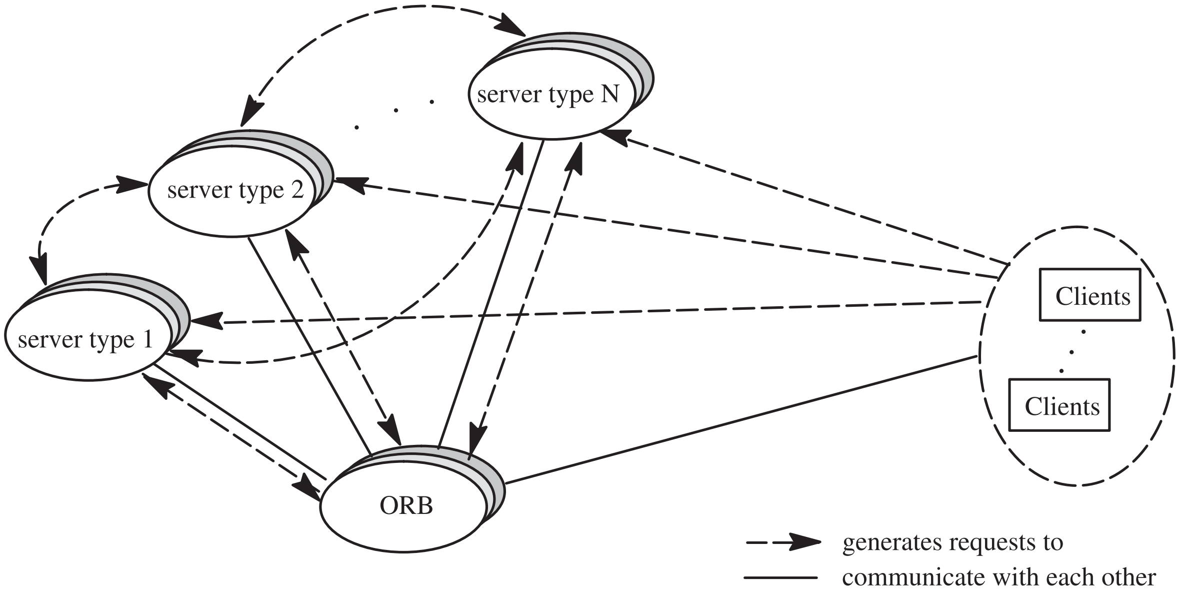


***Figure 1:*** **Server model of an enterprise-wide distributed WFMS**

on the currently entered state, and not on the previous history of entered states. The mathematical implication is that the residence time in a state - that is, the time the process resides in the state before it makes its next transition - follows a (state-specific) exponential distribution. Consequently, the behavior of a CTMC is uniquely described by a matrix $P = (p_{ij})$ of transition probabilities between states and the mean residence times $H_i$ of the states.
Let $\{a_i \mid i = 1..n\}$ be the set of *n* activities being part of a workflow type *X*. Let *X* be a workflow type without any subworkflows. The impact of subworkflows can be handled recursively and will be explained later. The control flow of an instance of *X* will be modeled by a CTMC where the states correspond to the workflow activities $a_i$. The state transition probability $p_{ij}$ corresponds to the probability that a workflow instance of workflow type *X* starts activity $a_j$ after it has completed activity $a_i$. The transition probabilities have to be explicitly specified by the workflow designer based on the semantics of the conditions between the workflow activities or observed from real-life business processes. The mean residence time $H_i$ corresponds to the mean turnaround time of activity $a_i$ and also needs to be estimated at workflow specification time. Finally, we add an artificial absorbing state *A* to the CTMC. This state represents the point when the workflow instance represented by the CTMC is terminated; the residence time of state *A* is infinity. Furthermore, we add a transition from every state of the CTMC that represents a termination activity of the workflow instance into state *A* with the transition probability *1*.
For workflow types with subworkflows, the subworkflows are initially represented by single fictitious states, i.e., each subworkflow is represented by a single state within the CTMC of the parent workflow. When workflows include parallelism, the parallel paths of the control flow are defined as subworkflows. In this case, the fictitious state represents all parallel subworkflows at once. As the mean residence time of the fictitious state, we use the maximum of the mean time until termination of all subworkflows within the fictitious state.
We derive the mean time until termination of a workflow type by the transient analysis of the CTMC representing the workflow type [Tij94]. In our model, a workflow instance of a workflow type terminates when the corresponding CTMC makes a transition into the absorption state *A*. So, the time until termination of a workflow type is equivalent to the mean time until the CTMC makes the first transition into state *A*. With the CTMC at hand, we are able to predict the expected load of workflow instances during their execution [Tij94] as shown in the following section.

# 3 Performance model

In this section, we present a performance model for a complete WFMS. We show how to describe the load for each workflow server type induced by the execution of a single workflow activity. We use this and the results from the transient analysis of CTMC presented in Section 2 for predicting the load induced by the execution of a entire workflow instance. Finally, we show how to predict the expected performance, i.e., sustainable throughput and expected response times of service requests, of the WFMS with a given system configuration.

## 3.1 Modeling activity-specific load

The execution of a workflow instance leads to the execution of a set of workflow activities. The execution of a workflow activity leads to the generation of service requests to different server types. Typically, the invocation of an activity leads to some initialization and termination load induced exactly once during the execution of the activity and to operational load induced continuously during the whole execution time of the activity. Therefore, we differentiate between the two following kinds of service requests.

- *Lump requests*. Lump requests are generated exactly once during the execution of the workflow activity. For example, if the activity corresponds to editing a text document, the activity generates a number of lump requests for loading the text document, updating a database to reflect the changed workflow state, etc.
- *Operational requests*. During the execution of an activity, operational requests are generated with a specific rate. For example, these are generated by the exchange of messages between workflow engines for synchronisation and migration of workflow instances, saving of intermediate versions of the currently processed documents, etc.

In the following, the matrix $(L^t_{sa})$ denotes the number of lump requests being generated for workflow server type $s$ when workflow activity $a$ is invoked during the execution of an instance of workflow type $t$. The matrix $(N^t_{sa})$ denotes the generation rate of operational requests for workflow server type $s$ during the execution of activity $a$.

### 3.2 Predicting the load induced by a single (sub-)workflow instance

To calculate the workflow load that one workflow instance generates on the several server types, we use the transient analysis of the CTMC presented in Section 2. To be exact, we combine the equivalent normalized CTMC [Tij94] and the already presented load matrices $(L^t_{sa})$ and $(N^t_{sa})$ to a Markov reward model (MRM). The feature of a MRM is that there are rewards for every state of the CTMC. To get the expected number of service requests by a single workflow instance, we calculate the expected reward earned until absorption [Tij94]. Let the matrix $(L^t_{sa})$ of lump requests and the matrix $(N^t_{sa})$ of rates of operational request be given for a workflow type without any subworkflows. Then, the expected number of service requests an instance of the workflow type $t$ generates at server type $s$ is given by

$$r_{s,t} = \frac{1}{\nu_t}\left[\sum_{a \neq A} N^t_{sa} \sum_{z=0}^{\infty} p^t_{\emptyset a}(z) + \sum_{a \neq A} \sum_{z=0}^{\infty} p^t_{\emptyset a}(z) \sum_{b \neq A, b \neq a} q^t_{ab} L^t_{sb}\right],$$

where $\nu_t$ is the maximum of the departure rates of the states of the CTMC representing workflow type $t$, $q^t_{ab}$ is the transition rate from state $a$ to state $b$, and $p^t_{\emptyset a}(z)$ is the taboo probability that the process will be in state $a$ after $z$ steps without having visited the absorbing state $A$ (starting in the initial state $\emptyset$).

The mean runtime $R_t$ of an instance of a (sub-)workflow of type $t$ is given by the mean time that the CTMC needs to enter the absorbing state for the first time, the so called first-visit-time of state $A$, and can be calculated by solving a system of linear equations [Tij94].

### 3.3 Incorporation of subworkflows

The expected number of service requests generated by an instance of a workflow type including subworkflows can be calculated recursively. For every state of the CTMC that represents a subworkflow or a set of parallel subworkflows, the entries $L^t_{sx}$ and $N^t_{sx}$ within the matrices $(L^t_{sa})$ and $(N^t_{sa})$ represent the lump requests and the rate of operational requests of the set $x$ of nested subworkflows. We approximate the number of lump requests $L^t_{sx}$ for a server type $s$ by the sum of the expected number of service requests generated by the parallel subworkflows.

$$L^t_{sx} = \sum_{y \in x} r_{s,y}$$

The rate of operational requests $N^t_{sx}$ is set to *0* for every server type $s$.

### 3.4 Incorporation of multiple active workflows

By Little's law, the steady-state number of active instances $N^t_{active}$ of workflow type $t$ is given by the product of the arrival rate $\lambda_t$ of new instances of type $t$ and the mean runtime $R_t$ of a single workflow of type $t$.

$$N_{active}^{t} = \lambda_t R_t$$

The server-type-specific request arrival rate of a single instance of workflow type $t$ is given by dividing the expected number of service requests to server type $s$, $r_{s,t}$, by the mean runtime of an instance of $t$. We obtain the server-type-specific request arrival rate $l_{s,t}$ of all instances of workflow type $t$ by multiplying $r_{s,t}$ with the mean number of active workflow instances.

$$l_{s,t} = N_{active}^{t} \frac{r_{s,t}}{R_t} = \lambda_t r_{s,t}$$

Finally, the request arrival rate $l_s$ to workflow server $s$ induced by all active instances of all workflow types is given by

$$l_s = \sum_t l_{s,t}.$$

### 3.5 Predicting response times for service requests

For predicting the mean response time of service requests, we model every server type as a set of $k$ M/G/1 queueing systems where $k$ is the number of server replications of the server type. We assume that the arriving service requests are uniformly distributed over all server replications. We thus compute the mean arrival rate of service requests for each M/G/1 queue by dividing the mean arrival rate of service requests for the server type by the number of servers $k$ of the server type. The mean service time of service requests and the second moment of the service time distribution are parameters that can be estimated by online monitoring.

## 4 Availability model

In this section, we describe our availability model. It is based on the workflow server model described in Section 2. Based on this model, we analyze the influence of transient component failures on the availability of the WFMS.

Our availability model is again based on Continuous Time Markov Chains (CTMC). Here, every state of the CTMC represents a possible system state of the WFMS. A system state of the WFMS is modelled as an $n$-tuple with $n$ being the number of different server types and each entry of the tuple representing the number of available workflow servers of a server type at one point of time. For example, the state *(2,1,1)* means that the WFMS consists of three different server types and there are *2* workflow servers of type *1*, *1* workflow server of type *2*, and *1* workflow server of type *3* currently available (the others have failed and are being restarted). When a workflow server of type $i$ fails, the CTMC performs a transition to the state where the corresponding value for server type $i$ is decreased by one. For example, the state $(X_1, ..., X_j, ...)$ is left when a workflow server of type $j$ fails, and the state $(X_1, ..., (X_j - 1), ...)$ is entered. Analogously, when a workflow server of type $i$ completes its restart, the value for server type $i$ is increased in the target state of the firing transition. The failure rates and the repair rates of the server types are the corresponding transition rates of the CTMC. Note that non-exponential failure or repair rates (e.g., anticipated periodic downtimes for software maintenance) can be accommodated by refining the corresponding state with non-exponential residence time into a chain of exponential states [Tij94].

With the CTMC at hand, we are able to calculate for every state of the WFMS its steady-state probability by solving a system of linear equations [STP96]. From these probabilities, we can then derive the probability distribution of the number of available workflow servers for each server type, and thus the server type's steady-state availability.

## 5 Performability model

In this section, we briefly sketch a performability model that allows us to predict the performance of the WFMS with the effects of temporarily non-available servers (i.e., the resulting performance degradation) taken into account.
Our performability model is a hierarchical model constituted by a Markov reward model (MRM) for the availability CTMC and the performance model presented in Section 3. The probability of being in a specific state of the WFMS is given by the availability model presented in Section 4. As state-specific rewards, we use a function that assigns to every state of the availability CTMC the mean response time of service requests of the WFMS in the current state. The steady-state analysis of the MRM delivers the expected value for the response time of service requests for a given configuration of the WFMS [STP96].

## 6 Summary and Outlook

In this paper, we have discussed three models to derive quantitative information about performance, availability, and performability of distributed workflow management systems (WFMS) configurations. These models form the core towards an assessment and configuration tool for enterprise-wide, large scale WFMS. We are in the process of implementing such a tool. The tool consits of four components: *mapping* of workflow specification onto the presented models, *calibration* by means of statistics from monitoring the system, *evaluation* for given input parameters, and the computation of *recommendations* with respect to specified administration goals. When the tool is to be used for configuring a completely new workflow environment, most input parameters have to be inetellectually estimated by a human expert. Later, after the system has been operational for a while, these parameters can be automatically adjusted, and the tool can the make appropriate recommendations for reconfiguring the system. For a first evaluation of our overall approach, we have defined a WFMS benchmark [GMW+99] and we are conducting measurements of various products and prototypes, including our own Mentor-lite system. These measurements will serve as a first yardstick for the accuracy of our performance assessment model.

## References


[BD97] T. Bauer, P. Dadam, A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration, 2nd IFCIS Conf. on Cooperative Information Systems (CoopIS), Charleston, South Carolina, 1997

[BD99] T Bauer, P. Dadam, Distribution Models for Workflow Management Systems - Classification and Simulation (in German), Technical Report, University of Ulm, Germany, 1999

[DKO+98] A. Dogac, L. Kalinichenko, M. Tamer Ozsu, A. Sheth (Eds.), Workflow Management Systems and Interoperability, NATO Advanced Study Institute, Springer-Verlag, 1998

[GMW+99] M. Gillmann, P. Muth, G. Weikum, J. Weissenfels, Benchmarking of Workflow Management Systems (in German), 8th German Conf. on Database Systems in Office, Engineering, and Scientific Applications (BTW), Freiburg, Germany, 1999

[HA98] C. Hagen, G. Alonso, Backup and Process Migration Mechanisms in Process Support Systems, Technical Report, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1998

[KAG+96] M. Kamath, G. Alonso, R. Günthör, C. Mohan, Providing High Availability in Very Large Workflow Management Systems, 5th Int'l Conf. on Extending Database Technology (EDBT), Avignon, France, 1996

[Nel95] R. Nelson, Probability, Stochastic Processes, and Queueing Theory, Springer-Verlag, 1995

[STP96] R. A. Sahner, K. S. Trivedi, A. Puliafito, Performance and Reliability Analysis of Computer Systems, Kluwer Academic Publishers, 1996

[SNS99] H. Schuster, J. Neeb, R. Schamburger, A Configuration Management Approach for Large Workflow Management Systems, Int'l Joint Conf. on Work Activities Coordination and Collaboration (WACC), San Francisco, California, 1999

[Tij94] H.C. Tijms, Stochastic Models, John Wiley and Sons, 1994