Research Article

Quality and Reliability Engineering International

(www.interscience.wiley.com) DOI: 10.1002/qre.1061

Published online 10 August 2009 in Wiley InterScience

Modeling Process Flow Using Diagrams

Benjamin Kemper,*† Jeroen de Mast and Michel Mandjes

In the practice of process improvement, tools such as the flowchart, the value-stream map (VSM), and a variety of *ad hoc* variants of such diagrams are commonly used. The purpose of this paper is to present a clear, precise, and consistent framework for the use of such flow diagrams in process improvement projects. The paper finds that traditional diagrams, such as the flowchart, the VSM, and OR-type of diagrams, have severe limitations, miss certain elements, or are based on implicit but consequential premises. These limitations restrict the applicability of traditional diagrams in non-manufacturing areas such as service and healthcare processes.

We show that a rational reconstruction for the use of diagrams in various disciplines regarding process flow boils down to a generic framework of elements, definitions of generic process metrics, and three classes of applications, namely the 'as-is', 'could-be', and 'should-be' analysis.

The goal is not to replace all currently used diagrams, but merely to discuss the role of diagram usage in process flow modeling. This paper provides an explicit framework that is unambiguous and flexible, and has the potential to serve as a guideline for the practitioner, in manufacturing as well as in service and healthcare. Besides, it may serve as a starting point to develop an ontology of business processes. Copyright © 2009 John Wiley & Sons, Ltd.

Keywords: rational reconstruction; process improvement; flowchart; value stream map; operations research

1. Introduction

In processes in manufacturing, service delivery organizations, healthcare, and elsewhere, one encounters a cluster of problems related to issues such as:

- Ensuring acceptable lead times.
- Matching capacity (in man-hours or machines, for instance) to workload.
- Minimizing productivity losses due to rework.

We are not referring here to problems that are solved by introducing exogenous remedies, such as replacing a manual process with a (partly) automated one. Rather, we are talking about dealing with the above-mentioned issues by optimizing the process itself. The reader may think of improvement actions such as:

- Increasing capacity at some workstations, reducing capacity at others.
- Implementing changes in queue management, such as replacing prioritization rules or replacing a 'push' discipline with a 'pull' discipline.
- Modifying or standardizing routing through the process.
- Reducing rework and iterations in the process.
- Replacing batch-wise production with a single-piece flow discipline.

We will call this cluster of problems the *process flow*. This paper discusses graphical techniques for process description, which facilitate the study of these problems in process improvement projects[1, 2]. Traditionally, the development of methods for such studies has been one of the pursuits of operations research (OR)[3]. Some of OR's roots are in the modeling and analysis of manufacturing systems (roughly from the 1950s onwards). Later, increasing emphasis was put on applications in data communication networks; see for instance the book by Kleinrock[4]. By now OR has become a mature scientific field that offers a substantial set of techniques addressing efficiency issues in both deterministic and stochastic networks, such as production and

Institute for Business and Industrial Statistics of the University of Amsterdam (IBIS UvA), Korteweg-de Vries Institute for Mathematics, Plantage Muidergracht 12, Room M 1.38, 1018 TV, Amsterdam, The Netherlands

**Correspondence to: Benjamin Kemper, Institute for Business and Industrial Statistics of the University of Amsterdam (IBIS UvA), Korteweg-de Vries Institute for Mathematics, Plantage Muidergracht 12, Room M 1.38, 1018 TV, Amsterdam, The Netherlands.*

†E-mail: b.p.h.kemper@uva.nl

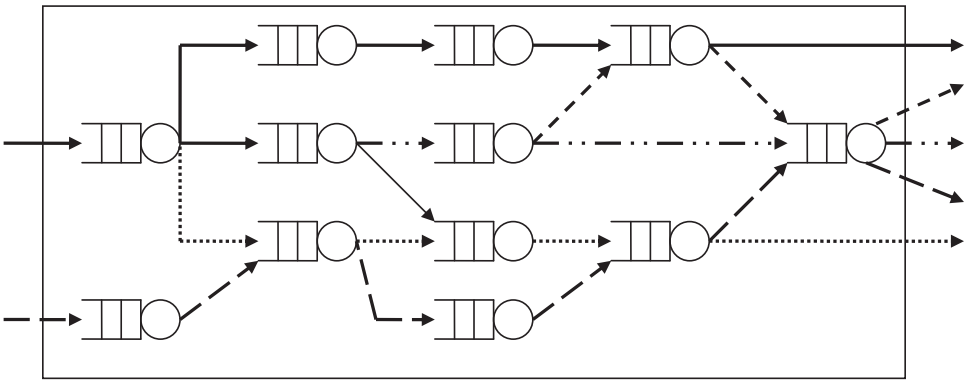

Figure 1. An OR-type diagram of a queuing system. Circles depict servers, horizontal stacks represent queues, and arrows represent job routes

service systems. A central role in these techniques is played by (mathematically oriented) methods for evaluating the performance of the underlying networks, and mechanisms to optimize this performance, such as planning and scheduling.

A more pragmatic approach is Lean Manufacturing (also referred to as Lean Thinking). This approach, which evolved from the Toyota Production System, was popularized in recent years and is now widely applied in the manufacturing, the service industries, healthcare, and beyond[5]. It prescribes a number of standard remedies for improving process flow, such as rapid changeover, 5S, pull/Kanban instead of push flow, and line balancing[6].

In both disciplines, the study and optimization of process flow are facilitated by diagrams. These diagrams are the topic of this paper. The traditional type of diagrams (used in the total quality movement, for example) is the flowchart. Its origins can be traced to Gilbreth's work in the era of scientific management[7]. This flowchart helps us visualize industrial and business processes and shows how jobs flow through a network of tasks and decisions. The flowchart does not display quantitative information about process flow, and does not depict servers. Although there seems not to be an authoritative set of symbols, in practice the more commonly used elements are depicted using standard symbols such as Chapin's basic and additional outlines[8].

In addition to flowcharts, operations researchers use a type of diagrams such as the one in Figure 1 (see, for example Tang *et al.*[2] and Altıok[3]). Whereas there appears to be a de facto standard for flowcharts, the symbols and structures used in OR-type diagrams seem to be *ad hoc*. Note the essential difference between this type of diagrams and flowcharts. While flowcharts follow jobs through a network of *tasks*, OR-type diagrams follow jobs through a network of *servers*.

In Lean Manufacturing a type of diagrams is used, which is called *value-stream map* (VSM)[9, 10]. The symbols given in Rother and Shook count as the de facto standard[11]. A VSM traces jobs through a network of stations, which can be workstations, queues/warehouses, and suppliers, see for example Seth and Gupta[12]. By employing symbols and the so-called *data boxes* it documents information about process flow, such as processing times, queue times, information flows, and queue disciplines (pull versus push)[13].

The above-mentioned types of diagrams for modeling process flow emerged and evolved in and through practice. Prescriptions given in textbooks and training manuals are pragmatic rather than precise, and tied to specific applications rather than generic. Moreover, the adoption of traditional quality engineering techniques from manufacturing in other fields, such as the service industries and healthcare, requires a clear, consistent, generic, and precise prescriptive framework for diagrams intended to model process flow. It is the purpose of this paper to present such a framework.

We will hardly go into the matter of which symbols to use—this remains a non-essential matter of convention. Instead, we aim to specify a generic list of elements of diagrams for process flow, such as tasks, servers, routing, and more. Next, we aim to specify properties of each type of element, which are relevant for the understanding process flow, such as cycle time, waiting time, capacity, and utilization.

Such an explicit framework has the potential to have substantial impact both on the analysis techniques focused on in academia, and on the development of practical guidelines. Our thorough analysis shows that currently popular types of diagrams have severe limitations, miss certain elements, or are based on implicit but consequential premises. A better articulated framework provides for more effective guidelines for the practitioner, and a unified and precise modeling language facilitates scientific mathematical studies of process flow, such as those carried out in OR.

Our approach comes down to a rational reconstruction of expositions given in the literature and current practice. Critically verifying the consistency of these expositions and confronting the given prescriptions with real-life applications, we identify a number of problematic issues with the traditional types of diagrams. In Section 2 we give an example and describe these problematic issues. Our solutions to these problems are contained in our framework, which are discussed in Section 3. Section 4, finally, presents a selection of applications. Section 5 concludes.

2. A grammar for process diagrams

We define a process as a system of activities that transform input into output. The input can be a request from a client at a bank, a patient in the hospital, raw material in manufacturing, or the output of a preceding process. Similarly, the output can

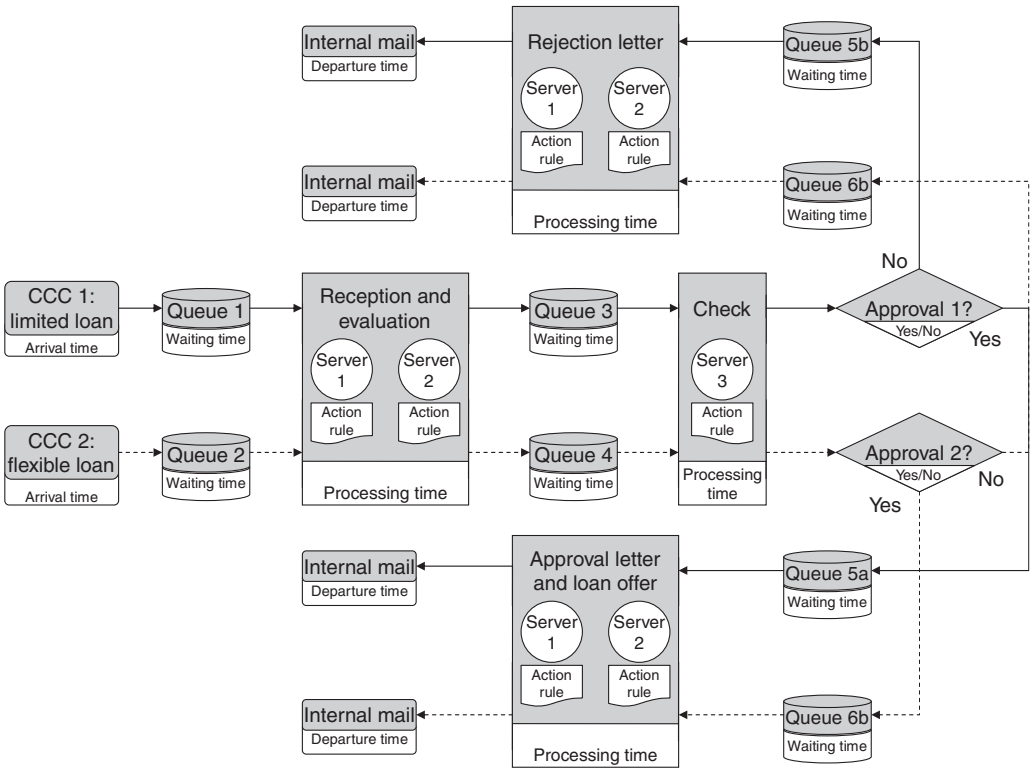

Figure 2. A back office process handling loan requests at a bank

serve as an input of a subsequent process, but can also be a completed service for a client, a cured or diagnosed patient or an end-product.

2.1. *An example*

To have a tangible basis for introducing our framework to model process flow, we provide an example here. The process we discuss refers to a back office of a bank which processes requests for loans. After a request is filed and checked, either a loan is offered to the client or the request is rejected. The back office department handles two types of private loans: a limited loan and a flexible loan. In Figure 2, the process diagram depicts the back office process as a system of activities. Solid and dashed lines indicate the flows of requests for limited and flexible loans.

A request for a loan from a client enters the bank at the Customer Contact Center (CCC). From the CCC the request is forwarded to the back office. After a job enters the back office process, it waits until it is picked by one of the employees. At this task, the request for a loan is received by the back office and the solvency of the client is checked. Next, the request is approved or rejected based on the outcome of the first task, by a server who is authorized to execute the loan approval. Approved jobs go to 'Approval letter and loan offer', where an offer for a loan is made. Rejected jobs go to 'Rejection letter', where a rejection letter is printed. Output of the final tasks is forwarded to 'Internal mail'. The internal mail facilitates the process of sending the documents to the clients.

Industrial and business processes considered as a system of activities consist of generic elements. Diagrams used to map process flow make use of generic symbols to depict these generic elements. Besides a unique symbol, these elements have *properties*. The properties of an element can be displayed in the so-called *data boxes*, known from the VSM-literature[11].

In the example we identify the following generic elements, to be discussed in more detail in the next section:

- *Connector*: A job creation process or job departure process. Both link to other processes. In the example, a filed request for a loan is an output of the process at the CCC and generates a job in the back office process at CCC 1 and CCC 2.
- *Route*: The arrows that display the route of a type of job from one activity to the following. In the example, jobs of the limited loan type follow the solid arrows and of the flexible loan type follows the dashed arrow.
- *Queue*: A station where a job spends waiting time, without an operation being applied to it, such as in Queue 3 and 4, where requests wait until they are checked.
- *Task*: A station where an operation is applied to a job. A task is executed by one or more servers, possibly in combination with a machine. In Figure 2 a server executes an approval procedure in the task 'Check', and with this task it singles out requests from insolvent clients.

- *Server*: An agent who executes a task on a job. In the example, two servers execute the task 'Reception and evaluation' of requests. Note that the same servers 1 and 2 execute all of the tasks 'Reception and evaluation', 'Approval letter and loan offer', and 'Rejection letter' during a shift.
- *Action rule*: A set of rules for the server, possibly in the form of a document. In the example, the action rule plans the capacity of servers 1, 2, and 3, schedules the different tasks for servers 1 and 2, and dictates to servers in which order jobs are to be picked from the queue.
- *Conditional routing*: A job status check that decides upon the next activity. Approved loan requests go to the 'Approval letter and offer' task and rejected loan requests go to the 'Rejection letter' task.
- *Job*: In the example the jobs that flow through the process are requests for loans.

With the help of a process diagram such as in Figure 2, we can trace a single job through the system. In the data boxes of the diagram, we display various quantities that are properties of the job as it flows through the process. For example, in the data box related to connector CCC 1, we can display the arrival time of a certain job. In the data box of Queue 1, we display the job's waiting time. Further, the data box of a task displays the job's processing time executed by a certain server. The data box of Routing 1 displays whether a loan request is approved, denoted by *Yes*, or rejected, denoted by *No*. We propose to call such a process diagram, in which the quantitative properties of an individual job are documented, a 'single-job flow diagram'.

Alternatively, in data boxes we could display aggregate statistics for a population of jobs. For example, in the data box related to the connector CCC 1, we can display the arrival rate of jobs of limited loan type. In the data box of Queue 1, we can display the average waiting time of this type of job. Further, the data box of a task can display an average processing time or a processing rate per time unit per server. The data box of Approval 1 displays an approval rate as aggregate statistic. We propose to call such a process diagram as an 'aggregate flow diagram'. Note that the statistics chosen in an aggregate flow diagram are not generic. One may choose to display statistics other than we suggest, such as the maximum waiting time per queue. Or, in case of a call center, one can choose even more complex ones, such as the average waiting time of jobs that wait at least 20 s.

2.2. *Shortcomings of current flowchart, VSM, or OR-type diagrams*

The flowchart, the VSM, and the OR-type of diagrams facilitate the study of process flow. Although, in essence, these diagrams have the same objective, they differ remarkably from each other in the inclusion of basic elements. For example, the flowchart and the VSM do not display servers explicitly, as in 'Server 1' and 'Server 2' at the 'Reception and evaluation' task in our example. Nor do these diagrams give information about which server executes what task and according to what schedule, as we display explicitly in the 'Action rule' of a server. The flowchart and VSM only deal with interchangeable servers and thus disregard differences among servers in authorities, skills, and experience. These server-specific characteristics are crucial information for optimal resource allocation. These diagrams, therefore, are not suited for modeling processes with multi-task employees.

The VSM does not explicitly inform us about the rejection rate due to conditional routing and the distinction between a single-job flow and aggregate flow is not made explicitly. Another drawback of the VSM is the limited use of the different queue handling or priority rules. In service or healthcare a first-in–first-out (FIFO) discipline is often not the optimal priority rule, and thus the possibility to model more complex rules for queue handling is important. For example, in data transfer over internet one chooses to handle the smallest jobs first in order to serve a large amount of small data packages first. In healthcare, emergency cases are treated with priority, and thus patient handling deviates from FIFO.

The OR-type diagram follows a job through a system of servers, instead of tasks as in a flowchart and VSM. For each server the corresponding queue is displayed. This diagram is widely applicable to queueing problems in OR, but it does not model basic elements such as tasks, conditional routing, and action rules.

Our proposed process diagram displays information about a job's (inter) arrival time explicitly in a data box of the *single-job flow diagram*, or an average waiting time of a population of jobs in a data box of the *aggregate flow diagram*.

Our diagram displays tasks, servers, and their corresponding action rules. It allows for the situation that servers are non-interchangeable (in authorities, skills, and experience), and therefore requires explicit capacity planning (which server does perform the task?) included in the action rule. It allows for the situation that servers are assigned to more than one task, and therefore requires an explicit server schedule (when is a multi-task server performing a task?) included in the action rule. Both situations are common in healthcare and service processes. The inclusion in our diagram of action rules (representing, among others, planning and scheduling) allows process optimization to take full advantage of differences among servers (in authorities, skills, and performance), as well as the possibility to allocate them to multiple tasks.

Our diagram has enough flexibility to accommodate the various forms of action rules found in industrial and business processes. Further, the distinction between single-job and aggregate flow diagrams is important, and therefore made explicit in our framework. Table I summarizes the contrasts between the flowchart, the VSM, and the OR-type diagram, and the proposed framework.

3. Detailed discussion of the elements in a process diagram

Although we tried to follow the flowchart and VSM literature in our choice of symbols, we do not prescribe the use of any specific symbols, as these are just a matter of convention. We discuss the elements in our generic framework in more detail, give some examples, and list properties for each of them. The definitions of the properties are in Tables II and III.

Table I. The difference per element between the flowchart, VSM, OR-type diagram, and the proposed framework

Characteristic of diagram	Flowchart	VSM	OR-type	Proposed framework
Server	Not displayed explicitly; tasks are the main building blocks of the diagram	Not displayed explicitly; tasks are the main building blocks of the diagram	Servers are the main building blocks, instead of tasks	Servers displayed explicitly
	No flexibility among servers in task-assignment and schedule (i.e. servers are interchangeable and tied to one task)	No flexibility among servers in task-assignment and schedule (i.e. servers are not interchangeable and tied to one task)	No flexibility among servers in task-assignment and schedule (i.e. servers are not interchangeable and tied to one task)	Are non-interchangeable; differ in skills and experience
				Can be allotted to multiple tasks
				Are accompanied by an action rule; schedule and queue handling
Single job/aggregate	Distinction not made	Distinction not made	Distinction not made	Distinction described and facilitated
Queue handling	Not displayed	Limited; mainly FIFO	Not explicitly displayed	Standard and non-standard queue handling rules are made explicit
Conditional routing	Displayed, but no percentage indicated	Displayed, but no percentage indicated	Not displayed	Displayed with percentage
Process flow metrics	Not included	Limited; not well defined nor standardized	Not included	Included and well defined

Table II. Metrics that are typically included in the data boxes of single-job flow diagrams

Metric	Definition	Element
Single-job flow diagram metrics		
Arrival time	The time a job arrives in the process	Connector
Departure time	The time a job departs from the process	Connector
Process throughput time	The total time a job spends in the process	Connector
Processing time	The time it takes a server to perform a task onto a job	Task, Server
Setup time	The time it takes a server to prepare a task for one or several jobs	Task, Server
Cycle time	The sum of processing time and setup time	Task, Server
Waiting time	The time a job spends in a queue	Queue

3.1. Connector

Connectors are the job creation and departure elements that demarcate the process under study. The generic symbol used for a connector is a rounded rectangle.

Examples:

- Job creation: Filed requests for loans are forwarded by the CCC to the back office of a bank; a downstream production process in manufacturing delivers semi-manufactures to the first workstation, or to the first queue related to this workstation; patients arrive at the front desk of a hospital with a referral from their general practitioner.
- Job departure: A letter possibly together with a loan offer is forwarded from the bank's back office to the internal mail; a manufacturing department forwards the end products to the shipping/expedition area; or a patient leaves the hospital when the treatment is finished.

Table III. Metrics that are typically included in the data boxes of aggregate flow diagrams

Metric	Definition	Element
Aggregate flow diagram metrics		
Arrival rate	Average number of job arrivals per unit of time	Connector
Departure rate	Average number of job departures per unit of time	Connector
Process throughput time distribution (average, standard deviation, percentiles)	Distribution of the total time jobs spend in the process	Connector, Process
Process throughput	Average number of jobs handled in the process per unit of time	Connector, Process
Processing time distribution (average, standard deviation, percentiles)	Distribution of the time it takes a server to perform a task onto a job	Task, Server
Setup time distribution	Distribution of the time it takes a server to prepare a task for one or several jobs	Task, Server
Cycle time distribution	Average of the sum of processing time and setup time of a task	Task, Server
Task throughput	Average number of jobs handled at a task per unit of time	Task, Server
Task capacity	Maximum possible throughput at a task given the availability of servers	Task, Server, Process
Task utilization	Ratio of task throughput and task capacity	Task, Server
Idle time	Average time a server is not processing a job or setting up a task, per unit of time	Server
Waiting time distribution (average, standard deviation, percentiles)	Distribution of the time a job spends in a queue	Queue
Ratio	Percentage of jobs that is routed to one of the several consecutive queues or tasks	Conditional routing

Typical metrics included in the data boxes of a single-job diagram are a job's arrival and departure time, as well as their difference, which is the job's process throughput time. Note that the process throughput time equals the sum of the setup times, processing times, and queue times that the job encounters in the process.

Typical metrics included in an aggregate diagram are the arrival rate (also known as workload or takt rate), the departure rate, and the distribution of the process's throughput times (or derivatives, such as the mean or certain percentiles).

3.2. *Task*

The generic symbol used for a task is a rectangle. Tasks are the operations performed by one or more servers, possibly in combination with a machine. For example, a solvency check of a client in a service process at a bank; an operation along the assembly line or the transport of a product in manufacturing; or an intake of a patient at the front desk of a hospital.

Typical properties included in a single-job flow diagram are processing times and setup times at the various tasks. The sum of the average processing and setup times at a task is the task's cycle time. Typical properties in the aggregate flow diagram are the processing time distribution and the setup time distribution (and derivatives such as means and standard deviations). The average number of jobs handled at a task per time unit is the task throughput. Given the availability of servers, the maximum possible throughput is the capacity at the task. The ratio of throughput to capacity is the utilization at the task. If utilization is below 1, the servers have a positive idle time.

Note that, contrary to what is often stated, the throughput does not in general equal the inverse of the mean cycle time, because the mean of an inverse does not coincide with the inverse of the mean.

3.3. *Server*

A server is an agent who executes a task on a job. The generic symbol used for a server is a circle. Since the server is a resource for processing time, the total number of servers and their planning and scheduling are important factors for the system's throughput.

Examples: An operator in manufacturing, an employee at a bank, but also a machine in an assembly line or a truck driver that facilitates the transport in manufacturing.

Note that most metrics under task are server specific. We do not assume the servers to be interchangeable; each server may have distinct characteristics such as skills, experience, or authorities. Further, servers need not be allotted to a single task.

3.4. *Action rule*

Action rules prescribe to servers what tasks to perform, when, in case of multi-task servers, to perform a task, and in which order to handle jobs. An action rule is typically a document that consists of planning, a schedule, and working rules. The generic symbol used for an action rule is a rectangle with a wavy lower edge attached to a server. It may also include a queue handling rule, such as FIFO, or *smallest-jobs-first*, in which latter case the job with the smaller expected processing time is handled first. Also it may contain a rule for priority handling, such as a *preemptive resume priority* rule. This rule prescribes that, on arrival of a priority job, a server is to quit the current job and handle the priority job first.

Examples: A planning of available time per server per task, a queue rule like FIFO or a priority handling rule that prescribes how to handle emergency patients in a hospital.

3.5. *Queue*

A queue is a station where a job waits for a server to become idle. The generic symbol used for a queue is a cylinder. In the single-job diagram, the job's waiting time is recorded in the queue's data box, while the aggregate diagram includes derivatives of the waiting time distribution.

3.6. *Job*

Jobs are the entities that flow through the process. The jobs in the processes we discuss in this paper move one by one or in batches from one task to the following task in the system. In the literature these types of flow are called single-piece flow and batch-and-queue, respectively. We do not discuss continuous-flow processes.

Examples: A loan request that flows through the back office of a bank, a car that flows through a part of an assembly line in manufacturing, or a patient that flows through one or more departments in a hospital.

3.7. *Route*

The route of a job is displayed as an arrow that connects every step in the process in a prescribed order. Different types of jobs can be depicted in one diagram. The generic symbol used for a route is an arrow, possibly with a certain pattern. In our example the routes are depicted by solid arrows for jobs of the limited loan type, and dashed arrows for jobs of the flexible loan type. The pattern or color of arrows is used to distinguish the route of different types of jobs.

3.8. *Conditional routing*

At a conditional routing, the path of jobs forks in one direction or another depending on a characteristic related to the condition of the job. The generic symbol used for a conditional routing is a diamond. In the single-job flow diagram in Figure 2 the rejected requests fork in the direction of the task 'Rejection letter' and the approved requests in the direction of the task 'Approval letter and loan offer'.

Examples: An approval check for loans, a quality check for products.

The single-job diagram shows which of the paths the job takes; the aggregate diagram shows the percentage of jobs taking each path.

4. Applications

Process diagrams, originally used as a design tool in information technology, nowadays visualize process flow and document process performance. They can display relevant performance metrics, based on the elementary metrics listed in Table III. A number of these are:

1. Performance metrics related to timeliness of delivery, such as process throughput time (or lead time). The diagram also shows the components of throughput time, namely processing times, setup times, and queue times.
2. Performance metrics related to iterations, such as rework rates and rejection.
3. Performance metrics related to the identification of bottlenecks in the process, such as a task's capacity and the throughput at the previous task upstream.
4. Performance metrics related to capacity planning, such as throughput and utilization.

The metrics can be determined by measurement, simulation, prediction from analytical methods (such as queueing-theoretic calculations), and typically a combination of these.

The process performance is determined by process settings. Process settings are those variables that are controllable and typically prescribed or set by management, such as resource planning parameters (number of servers, equipment, office rooms), scheduling parameters (schedule of tasks for an individual shift), and other settings (allocation of consecutive workstations, software, and databases).

The process diagram can be used to document, predict, optimize, and specify process performance. Most of these applications can be grouped in three application types, popularly known as 'as-is', 'could-be', and 'should-be' analysis.

4.1. *As-is analysis*

In most problems that regard the process flow, the first step in the analysis is to map the current process (the so-called *as-is*, *current state*, or *Ist-situation*), with the current process settings. Both the single-job flow diagram and the aggregate flow diagram can be used to document and visualize the current process performance. In business improvement programs, for example, the resulting diagram is helpful in identifying problems and improvement opportunities in the current process, or in prioritizing processes that are candidates for improvement projects.

In practice, it is not always possible to measure every step in the process under study. With the help of the statistics in an aggregate flow diagram and queueing models from OR, it may be possible to predict the metrics of the process steps that are not measured. For example, given the throughput of a task before a queue, and given the capacity of the task that follows the queue, one can predict from mathematical calculations the average waiting time.

4.2. *Could-be analysis*

A second sort of application of the diagram is to study a hypothetical 'what-if' process, based on OR calculus, simulation, or other analytical tools. One can make predictions about the process performance of alternative process settings or hypothetical circumstances. With the information from aggregate flow diagrams and corresponding settings, one can predict the performance of non-measured process settings with help of interpolation techniques, or one can make forecasts of process performance, based on for example next week's server capacity and expected demand.

Besides, one can optimize the process performance by experimenting with the settings of the process. In order to find the optimal settings one can use both empirical experimentation and simulation. One can for example experiment with various queue handling rules. Another example of such a process optimization is 'line balancing'. It means that schedules are adjusted to ensure that all tasks have a capacity slightly higher than the job arrival rate. One may for example change the schedule of the servers in the back office, so that (given the processing rate per server per task) the different tasks have equal processing rate per hour that is slightly higher than the rate of recorded requests at the CCC.

4.3. *Should-be analysis*

Third, the diagram is used to lay down the settings, and corresponding expected performance, of the future process. In the VSM literature these settings are documented in the so called *future-state* map[11]. The goal is to specify the settings so as to synchronize the process flow with upstream processes, downstream processes, or needs of the client. In a simple case of the future-state map, we specify the number of servers per task and the expected job arrival rate, as well as the predicted throughput times and utilization per task which result from these choices.

In the example of the loan request process in Section 2, one can set a throughput at every task based on the rate at which requests are recorded at the CCC. A next step might be to reschedule the servers' available times such that this throughput is just met. The should-be diagram translates the performance goal into operational definitions and process settings. It spells out the adjustments in the process and where to focus on in process improvement initiatives.

5. Conclusions

Commonly used tools such as the flowchart, VSM, and variants of the OR-type diagrams have similar functions, namely to model process flow. These variants can be seen as different dialects of a modeling language. Our framework is intended to provide a common, unambiguous, and flexible grammar for such language. Besides a grammar, a modeling language needs a vocabulary, which our *generic elements* provide. One step further would be the development of an ontology of business processes; see Dietz for a proposal[14].

In this paper, we identified various shortcomings of the flowchart, VSM and OR-type diagrams. They can be grouped under

- They are not generic enough; for example, they cannot accommodate multi-task or non-interchangeable servers, as are commonly encountered in service and healthcare processes.
- The elements that they consist of are poorly defined. The diagrams for modeling process flow emerged and evolved in and through practice, and as a consequence, the question which elements to include has not received much explicit consideration.
- They do not use well defined and unambiguous metrics, which is important for measuring and calculating process flow characteristics.

Our diagram deals with these shortcomings and is presented in a generic and well-defined framework.

Flow diagrams can be on the single job or aggregate level, referring to whether the metrics included in the data boxes refer to an individual job, or to a population of jobs. The distinction is important, and especially for aggregate process diagrams it is crucial to define the population of jobs considered, and in addition, one should specify which characteristics of the distribution one is interested in (for example, the average, the standard deviation, or certain percentiles).

The usefulness of flow diagrams consists in their facilitation of a variety of applications, which can be grouped in 'as-is', 'could-be', and 'should-be' analysis.

- The 'as-is' analysis is used to document and to complete information of the current process.
- The 'could-be' analysis is used to predict performance and to optimize process settings.
- The 'should-be' analysis is used to lay down the future process and to translate this goal in operational definitions and process settings.

These findings result in an explicit framework that has the potential to serve as a guideline for the practitioner, in manufacturing as well as in service and healthcare.

References

1. Montgomery DC, Woodall WH. An overview of six sigma. *International Statistical Review* 2008; **76**(3):329–346.
2. Tang LC, Goh TN, Lam SW, Zhang CW. Fortification of six sigma: Expanding the DMAIC toolset. *Quality and Reliability Engineering International* 2007; **23**(1):3–18.
3. Altıok T. *Performance Analysis of Manufacturing Systems*. Springer: New York, 1996.
4. Kleinrock L. *Queueing Systems Volume II: Computer Applications*. Wiley: Toronto, 1976.
5. Womack JP, Jones DT, Roos D. *The Machine that Changed the World: The Story of Lean Production*. Harper Perennial: New York, 1991.
6. Hines P, Rich N. The seven value stream mapping tools. *International Journal of Operations and Production Management* 1997; **17**(1):46–64.
7. Wren DA. *The History of Management Thought*. Wiley: Hoboken, 2005.
8. Chapin N. Flowcharting with the ANSI standard: A tutorial. *Computing Surveys* 1970; **2**(2):119–146.
9. McDonald T, Van Aken EM, Rentes AF. Utilising simulation to enhance value stream mapping: A manufacturing case example. *International Journal of Logistics Research and Applications* 2002; **5**(2):213–232.
10. Manos T. Value stream mapping—An introduction. *Quality Progress* 2006; **12**(6):64–69.
11. Rother M, Shook J. *Learning to See: Value-stream Mapping to Create Value and Eliminate the Muda*. The Lean Enterprise Institute: Cambridge, MA, 1999.
12. Seth D, Gupta V. Application of value stream mapping for lean operations and cycle time reduction: An Indian case study. *Production Planning and Control* 2005; **16**(1):44–59.
13. Braglia M, Carmignani G, Zammori F. A new values stream mapping approach for complex production systems. *International Journal of Production Research* 2006; **44**(18):3929–3952.
14. Dietz JLG. *Enterprise Ontology: Theory and Methodology*. Springer: Berlin, 2006.

Authors' biographies

Benjamin Kemper obtained a degree in econometrics at the University of Amsterdam, the Netherlands. He is a consultant at the Institute for Business and Industrial Statistics of the University of Amsterdam. He works on a doctorate project focusing on process flow optimization in service networks.

Jeroen de Mast obtained a doctorate in statistics at the University of Amsterdam. Currently, he works as a senior consultant at the Institute for Business and Industrial Statistics, and as associate professor at the University of Amsterdam. He has coauthored several books about Six Sigma, and is recipient of the ASQ 2005 Brumbaugh Award, as well as the 2005 ENBIS Young Statistician Award. He is a member of ASQ.

Michel Mandjes obtained a PhD in Operations Research and Applied Probability from the Vrije Universiteit, Amsterdam. After having worked as a Member of Technical Staff at KPN Research (Leidschendam, the Netherlands), and Lucent Technologies/Bell Laboratories (Murray Hill, NJ, United States), and several academic positions, he is now a full professor in Applied Probability at the University of Amsterdam. His research focuses on queueing theory and stochastic operations research, predominantly applied in the design of communication networks, but also in finance/risk, as well as in the production and service systems. He is the author of the recently published book Large Deviations for Gaussian Queues' (Wiley Interscience, 2007).