

Implementing a New International Paper Mill Efficiency Standard - Using Computational Correctness Criteria to Model and Verify Timed Events

Jari Veijalainen¹, Eleni Berki², Jari Lehmonen³, Pasi Moisanen⁴

Hasso-Plattner-Institut fuer Software Systemtechnik GmbH
University of Potsdam, Prof.-Dr.-Helmert Str. 2-3, P.O. Box 900460,
D-14482 Potsdam, Germany, j.veijalainen@hpi.uni-potsdam.de

²Information Technology Research Institute (ITRI)
P.O. Box 35 (Agora) FIN-40014 University of Jyväskylä, Finland
eleni.berki@titu.jyu.fi

³Wisdomus, Keskustie 20, FIN-42600 Multia, Finland
jari.lehmonen@wisdomus.com

⁴Institute of Measurement and Information Technology
P.O. Box 692, FIN-33101 Tampere University of Technology, Finland
pasi.moisanen@tut.fi

Abstract. In the currently used information system architectures of paper mills there is a huge number of signals generated at the lowest automation level by thousands of hardware sensors and other embedded components. These are accumulated and refined into a manageable number of semantically meaningful events by software layers above the physical and automation layer, stored into a persistent storage, i.e. log. (database). Some events are shown to operators that can acknowledge them through the user interface of the monitoring interface and issue control events. In this paper we discuss what formal properties the events and the log should have in order to be compatible with the proposed new standard paper line efficiency calculation rules and how a standard event log can be generated using the existing software and hardware components in a paper mill. Our main contribution is the standard log structure and an accompanying standard algorithm yielding comparable efficiency calculations in various environments. The inevitable heterogeneity between different paper mills is proposed to be overcome by a separate algorithm that maps existing event types into the standard ones and forms a standard log consisting of the instances. We evaluate our suggestions in a case study at a Finnish paper mill, where we have gone through the existing system, concrete logs and event types, and evaluated them against the needs of the new efficiency calculation standard and the standard algorithm implementing it.

1. Introduction

Paper mills are large industrial and business complexes [1, 2] where many information systems have been introduced for various purposes over time. Human operators use some of them for monitoring the paper production line and some other to control the line [3, 4]. There exist standard calculation rules for paper production line efficiency calculation that are currently widely used in Europe, the German Zell Cheming Merkblatt II/2, 1981. The *efficiency* is specified as a product of *time efficiency* and *material efficiency* over a certain specified time interval. The problem with these standard calculation rules is that they can be interpreted in different ways concerning how the time efficiency is measured, i.e. how *maximum time available (MTA)*, i.e. *production time*, *idle time*, etc. and *time not available (TNA)* are determined. Another problem is that the material efficiency is based on the weight of the produced paper. Only the saleable production is weighted on the scales though, whereas the weight of the produced paper on the machine roll is only estimated. In addition, the weight varies with the humidity of the paper and the humidity can greatly vary in different environments [5]. On paper machine lines with an offline coating the weight of the paper increases significantly due to coating after the reeler.

Background: The existence of several different, but at the same time similar, calculation rules of efficiency has led to confusion and a lack of comparability. The main targets of the new efficiency standard [6] are to clarify and harmonize the paper production calculation rules, to support the efficiency and capability development in paper companies and to improve the communication between the machine suppliers and paper industry. The new standard improves the accuracy in two respects: First, it specifies in a more *accurate* way which time intervals are considered as time not available, idle time, and production time. Secondly, it replaces the weight-based material efficiency with *area efficiency*. The area of the produced paper can be directly measured at the mill using various methods and it can also be measured while the paper is cut and wound into the customer rolls. Thus, one gets two independent directly measured values for the area. In addition, the area of the paper hardly changes due to humidity or other factors, in contrast to the weight. For these reasons, introducing area efficiency should increase the accuracy of efficiency calculations. Together, the changes in the standard calculation rules should shrink the error in efficiency assessment below one percent unit [5, 6].

Rationale: The weight-based efficiency figure, renamed production *quantity utilization factor*, will be used as secondary efficiency figure for finishing operations. This is vital, as paper is, in most cases, still sold in tons. The new efficiency standard determines what the calculation rules for efficiency are and how the baseline data for the calculation is obtained, but it does not determine in any way how the efficiency calculation should be organized within the information systems architectures of the mills. Our idea is that a viable approach to this problem is to define a data structure, standard *event log*, and a standard calculation algorithm based on its *formal correctness* properties. The algorithm and the data structure should, then, be correctly implemented in various mill information systems. As a result, the efficiency calculation results obtained applying the standard algorithm should always be

comparable for two different standard logs of the same or different production line. The standard event types in the log are, at the same time, the target structure for transformations that map the existing event types and instances in the real mill information system logs onto the standard ones. Thus, the standard log is the central means in creating *interoperability* between existing software components. After introducing the standard log and the standard calculation algorithm, the number of sources for inaccuracies in efficiency calculation results within a paper production line and between paper production lines should shrink. The differences that mainly remain are those that result due to different signal/event generation and handling rules at the lower mill information system layers and due to operator-controlled events, i.e. human decision making, in situations where no automated decision is possible. In addition, measurement errors at the lowest layer cause differences, but according to our studies, their impact is at most $\pm 0.5\%$ in the final efficiency figure [7]. Thus, reaching 1 % error margin in efficiency calculation requires that the above-mentioned errors comprise at most $\pm 0.5\%$.

Formal Methods for Problem Solving: We concentrated in providing the design for an adequate efficiency calculation component within a paper mill information system for the new efficiency calculation algorithm. System's adequacy in our case is related to the correct specification of the events monitoring and controlling, and the formal language in which these specification requirements are expressed. Diagrammatic and other techniques employed by structured methodologies in earlier phases of our work, though widely used and well known in software development, were not rigorous enough, resulting in ambiguities [5]. The main problem was that a certain shut down interval relevant for time efficiency calculation can be categorized into Time Not Available (TNA) or Time Available (TA) after 24 or 48 hours have passed after its start. This requires that the formalism used should be able to capture the length of time intervals and their allocation into different categories. Consequently, it was difficult to use non-mathematical methods to *precisely* capture the paper mill dynamic requirements for the new efficiency standard and to reason about them, particularly with automated proof systems. Therefore, a more formal specification of the systems models in general, and for the events' capture in particular, was needed.

From structured methods we used techniques such as Data Flow Diagrams (DFDs) and Entity Life Histories (ELHs), which suffer from ambiguities, which, in turn, may cause inconsistencies [8, 9, 10]. In some cases these techniques needed to be modified and extended in order to facilitate future implementations [8, 9, 10, 11, 12]. In contrast, computational methods placed emphasis on the development of an unambiguous, correct, complete and consistent, implementable design, through the use of rigorous mathematical notation and techniques. Validation and verification techniques are constantly required to ensure that the requirements specification is a correct representation of the system needs, and that both design artefacts and their correspondent implementation should be proved so, as to satisfy these needs at modeling and implementation level [13].

The modeling area that considers *time* and *change* is otherwise known as 'dynamic systems behavioral modeling'. This is notably an area, where dynamic properties and

temporal aspects of systems are studied. There exist comparisons of a plethora of modeling diagrammatic notations of different expressability and rigor, which manage to capture time and change with various degrees of success [14, 15, 16]. A recent characterization of this type of conceptual modeling, which also includes metamodeling aspects, has been carried out by [15], who mentions other reviews with similar and different findings, and comments on their effectiveness, contributions and limitations in specific application domains.

Formal Computational Methods for Dynamic Systems: With the exception of the dynamic theoretical models of computation such as Finite State Machines (FSMs), Backus-Naur Form (BNF), etc. and their generalizations, X-Machines and Extended Backus-Naur Form (EBNF) respectively, all ‘formal’ approaches to dynamic systems development [16, 17, 18, 19, 20] concentrate on the coverage of the modeling needs of particular application domains. The limited availability of automated tools makes their use even more difficult, especially when it comes to issues of test cases generation from specifications, computability, reverse engineering and re-engineering. Certain forms of deductive reasoning are often used for quality assurance, validation and verification, although a formal testing procedure is utilized in a limited fashion. Some formal methods such as process algebras for Communicating and Sequential Processes (CSP) and Calculus of Communicating Systems (CCS), which are also used to model time change and concurrency, and some of their extensions such as Timed CCS (TCCS) involve automated tools more than others such as Z and Vienna Development Method (VDM), which are used to model sequential systems [21, 22]. The extensions of the latter though, such as Z++ and Object Z seem to have wider applicability for domain specifics. Notations that describe a system’s dynamic behaviour well are Petri-Nets (used in many cases to also model concurrent processes), Statecharts and Finite State Machines; these are model-based diagrammatic formalisms with adequate textual visualisations of abstraction, too [16, 17, 18, 19, 20, 21, 22]. Moreover, Regular Expressions, Context Free Grammars and Backus Naur Form offer rich text descriptions and algorithmic steps to express events occurrence with more clarification details. Most of these formalisms, though, and other real time structured and semi-formal methods [23, 24, 25] do not address the real time needs of the efficiency calculations in paper mills. There, we need not only to model the order of events, but also duration between events, and in such a case none of the above formalisms alone would be strong enough to express the complete requirements unambiguously, correctly and consistently among different levels of abstraction. As can be seen in the next sections, the paper mill environment is highly dynamic in events occurrences, and with the new standard efficiency calculations, we needed to intuitively use a variety of the above formal computational techniques and sometimes a combination of them.

The paper is organized as follows: We first discuss the overall paper making process and show the places in the process where the baseline data is gathered for the efficiency calculations. We subsequently refer to the collection and identification of event data found in the real paper mill environment and modeled as Finite State Machine and State Transition Matrix (Appendix A) in previous stages of the research work [5, 7]. We then define the *event log* as a *finite sequence of event instances* based

on BNF abstract typology; we also define its *correctness criteria*, and show that the efficiency calculation can be performed based on the assumed criteria in a single pass for any query. We finally discuss the *implementability* of the standard event log based on existing real events and problems in real systems, such as local clock drifts at different components, missing events and so on. We close the paper with conclusive remarks and future development issues on the results of our work and on the use of the particular computational methods as thinking and modeling tools for capturing the event data in other paper mill environments for the realization of the new international efficiency standard. We briefly provide a critical appraisal on the opportunities and difficulties encountered by the application of formal computational methods in the real environment of the Finnish local paper industry.

2. Paper Production Process and Calculation Rules

Paper machines residing in paper mills produce paper. The material input to the paper making process is a water suspension containing one percent of wood fibers, fine materials of wood, chemicals, and most often also mineral fillers. Within the paper machine, the solid contents of the suspension are elevated to that of the paper that is typically 90-96 %. Dewatering takes place by free dewatering, suction and pulsing of the wet web on forming section, pressing of the web on press section, and by heating the paper web by steam on drying section.

The speed of the paper line is typically 1000-2000 m/min (60-120 km/h). The speed is dependent on many factors, such as the machine and paper grade produced. The width of the wire is typically 5-8 meters. The ready paper is wound on a machine reel. At our case line a full reel it was ca. 6 m wide and contained typically ca. 50 km paper, i.e. it was produced in less than an hour. Its area was, thus, roughly 300000 m², i.e. 0.3 km².

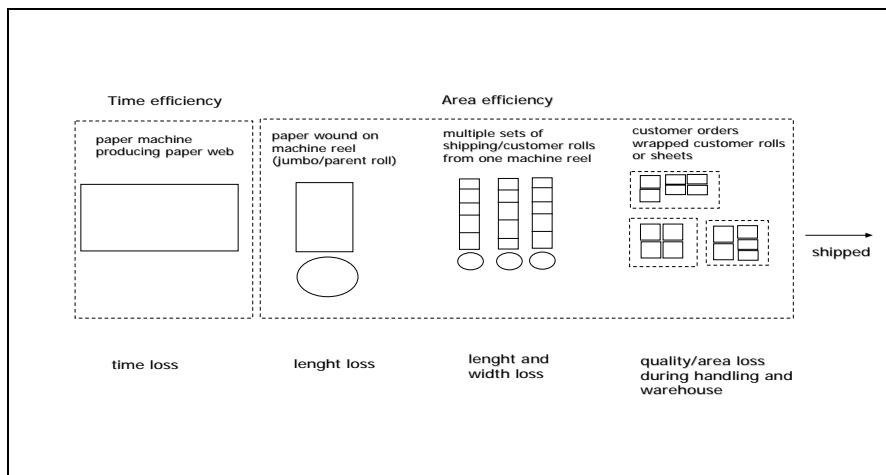


Fig. 1. A schematic view on a paper production line

The ready machine roll is subsequently cut into smaller rolls, called *customer rolls*. Customers have ordered the latter and each of the rolls must have exactly the width specified by the customer. The customer rolls are finally wrapped and weighted on a scale. After that they are moved to storage where they will wait for the transport to the customers (see figure 1). According to the new standard proposal, efficiency calculation for a paper production line is as follows:

$$\begin{aligned} \text{Overall efficiency} &= \text{Time efficiency} * \text{Area efficiency} \\ \text{Time efficiency} &= \text{Production time} / \text{Maximum time available} \end{aligned}$$

Time efficiency is defined as a ratio of the length of the *production time* to *maximum time available*, during a certain calendar time interval (typically a mill day, a month, a year). Production time is determined using the combination of the states of the machine chest pump (on/off), vacuum in the wire suction (on/off), and the paper machine reeler (paper wound/paper not wound on the reel).

$$\text{Production time} = \text{maximum time available} - \text{shut downs} - \text{idle time}$$

The *maximum time available* = *calendar time* – *time not available*, is, in practice, a fraction of calendar time. During idle time, stock is on wire, but no paper is wound on the reeler because of tail threading, break, or maintenance work.

$$\text{Area efficiency} = \text{Area of saleable production} / \text{Calculated average maximum produced area}$$

Area efficiency is calculated from paper machine reel onwards. The data for calculating area efficiency is available when the customer rolls are ready at winder. The area of returned production from outside the mill gate is reduced from saleable area. Rewinder salvage is added into the saleable area.

$$\begin{aligned} \text{Calculated average maximum produced area} &= \text{Paper length at paper machine reel} \\ &\quad * \text{Average maximum trim width at winder} \end{aligned}$$

This represents the area of the potentially saleable paper that the machine has produced during a certain time interval (mill day, month, year). The length above should be *accurately measured*. Using the average maximum trim width above might also cause a slight error, because it might deviate from the actual trim width used

Notice that the time efficiency, as defined above, does not tell what fraction of the calendar time the machine is running, nor the speed with which the machine is run during production, nor how much paper was produced by the machine during a certain period of time. The same holds for the area efficiency. The overall efficiency is a ratio between zero and one (hundred percent) and it tells primarily how many per cent of the potential maximum output the paper production line was able to produce during a given time interval.

3. Event Log as a Finite Sequence of Event Instances

When calculating the overall efficiency, i.e. the product of the time efficiency and area efficiency, one has to collect the baseline data from the same query time interval $[t_1, t_2]$ (the same mill day, month, year). This is an important requirement when organizing the data for efficiency calculations. The overall schematic architecture for the efficiency calculation is depicted in Fig. 2. The idea is that base line data is gathered into the *standard event log*.

An event log L is a finite sequence of event instances e_i , $i=1, 2, n$ and $n \in \mathbb{N}$, where $L = e_1 e_2, \dots, e_n$. A new event instance is always added at the end of the log. The insertion order is denoted by \ll below and called *log order*. In the mill environment, various sensors within the paper machine, human operators, cutters, wrappers and scales, continuously generate the event data (see also Appendix A). The log, thus, grows constantly and needs a generic and effective method to accurately and completely capture the event data and refine it for the calculations to be used in the new international efficiency paper mill standard. The efficiency queries are submitted when there is a need for them, or they are periodic.

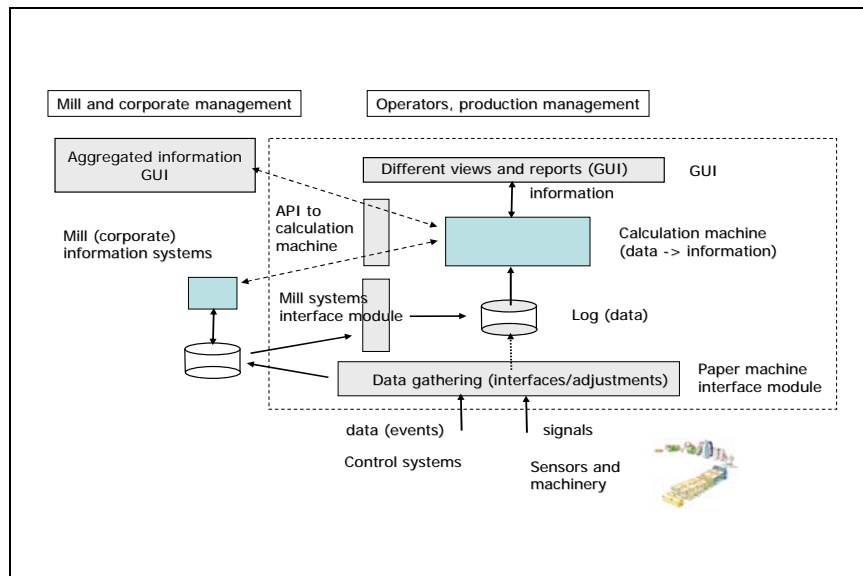


Fig. 2. Proposed overall architecture for paper machine efficiency calculation

There are many event types that are needed in monitoring and controlling a paper machine. In our real case environment of the Finnish paper mill, over 100 different event types were encountered and categorized as *internally-triggered*, *externally-triggered* and *time-triggered* [5]. Only a small portion of these is needed for efficiency calculations. Event sequences consisting of events of these types, modeled with finite state machines notation as deterministic and non-deterministic automata, are formally documented in the *time state event transition matrix* of Appendix A,

where a '?' indicates a missing requirement/event transition. We assume that only those absolutely necessary for this purpose are gathered into the standard event log in Fig. 2. Based on these, the efficiency figures defined above will be calculated upon request from applications or human users mentioned at the top of Fig. 2, namely mill and corporate managers, paper machine operators and production managers.

According to Turski in [26], formalizing the properties of an application domain is a challenge, because there are "two fundamental difficulties involved in dealing with non-formal domains (also known as 'the real world'). 1. Properties they enjoy are not necessarily expressible in any single linguistic system. 2. The notion of mathematical (logical) proof does not apply to them". In general, the use of computational methods for designing a system offers quality assurance regarding the development of unambiguous, consistent, correct and verified mathematically-proven specifications. That being the reality, a rigorous and integrated method based on formal specification rules must provide an appropriate syntactic and semantic structure to model the dynamic requirements of events occurrence and capture some or all of the features of specification and further computation. This can be achieved in terms of a specialized mathematical (therefore sufficiently generic) structure that can computationally express the richness of events specifications, considering implementation needs of further computer-based and hardware-based activity. The following section demonstrates the application of that structure.

3.1 The Event Structure in BNF Form

We define that an event in the event log has the following Backus-Naur Form (BNF) structure:

```

<event> ::= <ts><sid><et> %et = ev type, ec =event
<et> ::= <ec>["beg"|"end"<ts><sid>] % category
<ts> ::= integer; %time stamp in seconds from 1.1.1970
<sid> ::= <identifier> %event source identifier
<ec> ::= % time-related ("shutdown"|" personnel_na"<sh_reasons_pna>)|
"personnel_av"<sh_reasons_pa> ) | %no stock on wire, or no vacuum in the
%wire suction
"idle,"<idle_reasons> |
"production" %paper wound on reeler
<sh_reasons_pa> ::= "planned"<"technical"|"production">|
"unplanned"<reason_for_unplanned>
<sh_reasons_pna> ::= "holidays"|"work_agreements"|"strike"
<reason_for_unplanned> ::= .. %ca.10 codes,
<idle_reasons> ::= % stock on wire, no paper wound on reeler
"start_up"|"Technical/paper technical work"|"break"|"grade-change"
<reason_for_unplanned> ::= "Technical/paper technical work"|"Material shortage"|
"Lack of tambours"|"Repair"
%area related
("mr_ready"<mr_id><width><length>)|
("cr_saleable"<cr_id><width><length>)|
("cr_rejected" "<cr_id><width><length>)
```

<mr_id>::<identifier>
<cr_id>::<identifier>

If the event category of e is time-related, the event type contains *beg* or *end*. In the latter case the first field pair $[ts, sid]$ contains $e'[ts, sid]$ that is of the same category as e and $e' \ll e$ holds (i.e. it contains the identity of the event e' that starts the interval e closes). If the category of e is *mr_ready*, the parameter list contains values for *mr_id* (unique identifier of the machine roll), *width* (maximum trim width for the production line) and the length of the paper on the machine roll. For saleable production customer roll category events the parameter list contains *cr_id* (unique identifier of the customer roll), *width* (width of the customer roll) and length of the paper on the customer roll. The identifiers are all character strings, max ca. 20 characters long.

Let e be an event instance of the above structure. We refer to its fields by $e[x]$, where x occurs above within BNF-brackets $\langle \rangle$. Every event e has a *time stamp* $e[ts]$, whose denotation is explained later. We assume for the moment that the timestamp an event carries refers to the global system time (or wall clock time) when the original event (such as stock on wire signal, break, shut-down start event) was detected/generated by a sensor, human being, or some information system component. We assume for simplicity, and without loss of generality, that a timestamp is a number that tells how many seconds had passed from 1.1.1970 00:00:00 GMT to the occurrence of the original event. This is transformable to year:month:day:hour:minute:second in an *unambiguous* way when the time zone GMT \pm X hours is given, and vice versa.

The timestamp order is denoted by $<$; $e_1 < e_2$ iff $e_1[ts] < e_2[ts]$, where the latter order is the order of natural numbers. We also assume for the moment that all clocks in various hardware parts of the paper mill are accurate, i.e. do not drift, and that transmission and handling of events happens in no time. Thus, whenever an event is generated, it is appended immediately into the central standard event log in Fig.2. Furthermore, under these assumptions, the log order \ll and the timestamp order of the events, $<$, coincide in L.

Events related to time efficiency must carry the timestamp and further parameters indicating what time interval is beginning or ending. The objective is to record the beginning and the end of each of the above periods into the event log and calculate out of them, for the desired query time interval $[ts_1, ts_2]$, the combined production time and the Maximum Time Available MTA = production time + start-up time + grade change time + Technical/ Paper Technical Work time + break time + shut down times not exceeding 24/48hours with Personnel Available PA.

We call the target intervals in the log above *episodes* of category X. We assume that each episode j of type X has a *starting event* $e[ts, sid=j][ec=X][beg]$ and a *closing event* $e[ts, sid=k][ec=X][end][j]$ Event identifier j is unique within the log L. We assume that it is obtained e.g. from the timestamp the event carries, $e[ts]$, and from the unique identifier $e[sid]$ of the component generating the event, i.e. $j = e[ts, sid]$. j is an identifier of the episode. If there exists $e[beg]$ but no $e[end]$ for some event e_j , we say that the *episode j is open*. The area-related event types are *single*. This means that

instances do not have *beg* or *end* fields and that each occurrence has a unique identifier.

3.2 Correctness Criteria for a Temporally Correct Log

The temporally correct log is one that can be used by a standard calculation algorithm to calculate the efficiency figures. Further, it acts as a correctness specification for the algorithm that transforms a real log into the standard one.

Let $<$ denote the timestamp order and $<<$ the log order of L . Then, a *temporally correct log* L has the following correctness criteria:

- 1) The only log modifying operation is insertion of an event at the end of the log (the log cannot be modified later in the middle; result stability).
- 2) Any event instance e_j occurs at most once in the log (i.e. there are no duplicates).
- 3) All events in the log are in the ascending order according to the time stamp order $<$ (if the time stamps coincide, then $<<$ is applied).
- 4) If $e[ts, sid=j][ec=X][beg]$ and $e'[ts, sid=k][ec=X][end][j]$ occur in L , then $j < k$ (i.e. an episode must start before it ends).
- 5) At most one time-related episode can be open in the log i.e. if there is in the log $e[ts, sid=j][beg]$ but no $e'[end][j]$ then it must hold for all time-related events e'' : $e''[end] << e[ts, sid=j][beg]$ (that is, the log exhibits a unique time-related state at any point in time).

Based on the above criteria, it is easy to see that the different categories of episodes, i.e. time intervals, occur one after each other in a *correct* log and that only the last episode can be open. The time efficiency number can be calculated for any query time interval $[ts_1, ts_2]$ by one scan starting from the first event e from the beginning, whose time stamp is equal to ts_1 or that is larger than ts_1 . Then, the algorithm sums up the durations of each production time interval between $[ts_1, ts_2]$ and the duration for shut downs, brakes, grade changes, etc. i.e. MTA during $[ts_1, ts_2]$. The duration of an episode (in seconds) is directly $e'[ts] - e[ts]$ for any episode that starts with e and ends with e' .

3.3 Algorithmic Steps and Event Determination in a Paper Mill

For the area efficiency somewhat similar principles hold. The machine reel becomes at some point in time ready. At that point in time the corresponding event $e[ec=mr_ready]$ is generated. The *calculated average maximum produced area* during time interval $[ts_1, ts_2]$ is the sum of the areas of $e[ec=mr_ready]$ events during $[ts_1, ts_2]$. The same holds for the customer rolls. The algorithm calculating the efficiency only needs to calculate how much saleable production became ready during $[ts_1, ts_2]$. In addition, it needs to know which rolls were once recorded as saleable production, but later rejected by a customer. Rejection is a single event that causes the area of that customer roll to be deducted from the total saleable area accumulated during $[ts_1, ts_2]$. These rejection events get a time stamp within that mill day they were recorded as

rejected at the mill information system. The last step is to calculate the quotients and multiply them.

Notice that there might be open episodes when the query interval ends and the beginning might be in the middle of an episode. What is the right way of handling them? For area efficiency it is not a big issue, because the machine reel or customer roll becomes full or ready, at certain point of time, and either it is counted in or ignored [5]. For time episodes the problem can be solved in two ways: Because only one episode can be open at the end, the calculation algorithm could consider the open episode of category X to end at t_2 and add the fraction of the duration of the episode into class X . Or the algorithm could ignore an open episode at the end, i.e. leaving it out of any summation process. At the beginning, the duration of the partial episode $[ts_1, e[ts][ec=X [end][j]]]$ is summed into category X time.

The previous is a real problem for daily efficiency figures, but not so for monthly or yearly figure. There is, however, one complication. For shut-downs where personnel is available, there is a threshold of 24/48 hours that determines, whether the shut down time period belongs to maximum time available, or to time not available. The shut downs exceeding 24 or 48 hours are counted as part of time not available, but shorter are counted into maximum time available. In these cases the algorithm could consider that the shutdown ends at t_2 and then decide to which category (maximum time available/time not available) the beginning of the shut down time period belongs, if it is at the end. For the partial episode at the beginning, whose final category is known, this is not a problem.

All in all, if the log fulfilled the properties established above, a one-pass algorithm can always calculate the efficiency number. Further, the starting point of the scan can be determined e.g. by using binary search or index, it can do it for any query interval $[ts_1, ts_2]$ by starting the full scan at an event closest to ts_1 and ending it at closest to ts_2 . As a consequence, the algorithm only needs to scan the entire log L if interval $[ts_1, ts_2]$ covers L , otherwise only a portion.

4. Implementation Issues: Time Stamps and Set Signals

The time stamp order in the log is challenged in real life for many reasons. First, an event might be lost, so that it will never reach the standard log on its way from source to the log. This can be considered as an infinite delay in transmission. On the other hand, an event might be duplicated (cf. (2)). This is basically easy to handle algorithmically, by cleaning all but the first replica from the log, or prohibiting insertion of replicas, before using the log for efficiency calculation. Cleaning requires that the events indeed have a unique identifier.

Further, there are many subsystems in a mill information system and many processors with their own clock. The components are connected with each other through a mill network. This and the various software layers have different latencies when

processing and sending event-carrying messages to the subsystem that hosts the real standard event log. Thus, in the log order $e \ll e'$ might hold, but $e[ts] < e'[ts]$; e' came “too late” or e came “too early” to the standard log. These timing problems can lead to the following anomalies: an episode ends in L , although it did not start in L (cf. (4)); or it ends before it starts (cf. (3)); two or more episodes can be open in L at the same time (cf. (5)).

To correct the latter anomalies and make the standard log usable for the standard calculation algorithm, one needs to violate at least one of the correctness criteria above. For the starting and closing events of an episode that are in a wrong order, one can push the start event as far towards the beginning of the log as the correctness criteria 3) allows. If the start event is missing from the log, it can be generated into it immediately after the closest end event that makes 5) -and hopefully 3)- true. If the timestamp of the start event is contained in the end event of an episode, the start event can always be generated and correct duration is possible to calculate, even without rectifying the log. If another unique identifier is chosen, this would not work. Should an end event $e[\text{end}]$ be missing from the log, it can be generated based on the start event. It can be placed in front of the next start event e' of different time episode category so that 3) and 4) hold. One can use e' 's time stamp in e . As we see, in most cases the property 1) must be violated when correcting the log. This is not severe, fortunately, because the four further properties are those needed to guarantee that the standard algorithm works correctly.

Drifting clocks would cause a systematic violation of the log correctness, if the drift were “large enough”. The latter is in relation to the average length of the interval between events. If the drift is 1 second and events occur every 10 seconds, the event disorder due to the clock drift is observable perhaps in 10 % of the events. But if ten events occur per second, then the clock drift of 1 second manifests itself as event disorder for practically all events e the drifting clock is used to generate the time stamp for. This resolution trade-off makes the clock drift problem easier than in a general case (see reference [27]), but it still needs further attention. In the case study environment the clock drift was not considered a big issue, but changing between summer and winter time (1 hour difference) had caused problems.

The log can be rectified by enforcing criterion 5), but the time stamp order of the events 3) will most probably be violated. This would require the modification to the standard calculation algorithm. Another possibility is to update the erroneous time stamps and then rectify the log, before any standard calculation is performed on it. This requires that the drift is measured or estimated with a high enough accuracy. In our case mill the time stamps are shown to the operators with a minute resolution, although the system records seconds. This hints that acceptable clock drift might be seconds or even tens of seconds. The time stamp modification approach requires the definition of log equivalences. Log L_1 and L_2 are *efficiency equivalent*, if $EC(L_1, [ts_1, ts_2]) = EC(L_2, [ts_1, ts_2])$ for any time interval $[ts_1, ts_2]$, where EC is the function the efficiency calculation algorithm implements. This is rather hard to achieve if one changes the time stamps of events and rearranges the log. L_1 and L_2 are *episode equivalent* if they contain the same events, except that start and end event time

timestamps of some episodes might have been increased or decreased by the same constant. The idea is that the episode durations are the same in L_1 and L_2 , but their relative place in time stamp order might change. The usefulness of these log transformation concepts is for further study.

Let us finally mention something about the mappings between the current information systems and the proposed standard log based efficiency calculation. In the real case environment the system is already now event-based and almost all the information needed in efficiency calculations is contained in the existing events. We had to set up three signals, as follows:

- a=operational vacuum in the wire suction,
- b= machine chest stock pump is on,
- c=paper wound on reeler.

Based on the combination of these, events are defined: $\neg a \wedge \neg b \rightarrow$ shut down time (begins when condition becomes true, ends when $a \wedge b$), $a \wedge b \wedge \neg c \rightarrow$ idle time (begins, when condition becomes true, ends when $a \wedge b \wedge c$ or $\neg a \wedge \neg b$), $a \wedge b \wedge c \rightarrow$ production time (holds when condition true).

What is needed in terms of event transformations is a structural mapping from the current events to the abstract events as defined above in 3.1. We have at least two further options: either to specify a DTD for the actually implemented events or define a relational scheme. The real standard log would be a sequence of DTD-conform records and the calculation algorithm should be tuned for these. The implementation of the persistent log will most often be based on a commercial relational database. This favors the relational schema solution. Notwithstanding, these options are for further study and in the preference of the software programmers.

Finally, how can we cope with the problem of changing time category of an episode alluded to above. In log-based approach the problem appears basically so that an episode would begin with a shut-down event, but ends with an TNA ending event. This is not allowed in a standard log. The remedy is to add to the standard log an ending event for the shut-down interval immediately after its beginning (both with the same time stamp), and after them the beginning event of the TNA interval, again with the same time stamp as the shut-down events, when the fate of the shut-down is clear. It is easy to see that the log remains correct after this transformation. Thus, the standard algorithm finds all the (non-overlapping) beginning and end events for the episodes and calculates the correct durations (for the shut-down zero and for the TNA from the TNA beginning $ts =$ shut-down beginning ts to the TNA end).

Violating 1) above leads mainly to the problem that the result of the efficiency calculation for the same time interval $[ts_1, ts_2]$ and the same log can be different at different times. Therefore, there should be "a maximum allowed modification time" (MMT) specified after which the log cannot be changed any more. Based on this, we can show that the efficiency calculation results are repeatable for the beginning of the log, up to the events with time stamps smaller than now-MMT. What MMT is suitable and whether this could be globally enforced is for further study.

5. Conclusions and Future Development

There exists a new efficiency standard calculation draft for paper mills. In this paper we presented the basic principles for the event log-based implementation of the standard. In so doing, we used a formal specification approach that conforms to algorithmic steps with correctness criteria, based on the properties that characterize event-based systems of dynamic nature. The approach exposed in this paper, which is part of our project's proposed solutions for the design of the software-based information system, firstly defines the structure of the events and correctness criteria for an abstract event log, standard log. Based on them, it was possible to show that a one-pass algorithm can calculate the efficiency result for any query interval. We then analyzed how the log could be implemented in a real environment, where the real log can contain duplicated or missing events, wrong time stamps, and disordered events due to various latencies. We showed that the correctness criteria can help to rearrange, clean and complement the standard log so that the one-pass algorithm still works. The standard log is at the same time a correctness specification for the algorithm that transforms a real log into the standard one. To which extent this is applicable and generalizable in other than paper mills real environments is for further study. Since, though, this method demonstrates the strengths of a property-oriented abstraction to specify paper mills requirements, makes it possible for the method to have applicability in other paper mills in the world for the adoption of the new international efficiency standard; and possibly in many other similar specialized application domains that are event-based, via the generic abstraction it possesses as a computational model.

Throughout this paper and throughout our project work, *formal computational methods* were widely used to dynamically express paper mill subsystems properties, algorithms, data, events and functions in a rigorous mathematical way, which is not ambiguous. The family of computational methodological notations we used was *model-oriented*; it included diagrams such as FSMs, X-Machines and *property-oriented* structures such as BNF, which is based on text reasoning descriptions for events modeling. This combination of computational methods has so far been a wise choice for our application domain (paper mill environment) since the major drawback of many other formal methods based on mathematical notations lack structure, which makes it difficult to diagrammatically contribute to the documentation of large complex systems. Formal methods also tend sometimes to ignore the business aspects and they offer minimal user involvement and limited automated tool support [28]. This is due to the complexity of the mathematical notations, which, in some cases require specialist knowledge. This, in our case was sometimes considered a problem, as we needed to guarantee the active participation of the paper machine operators, local paper production managers and the involvement of corporate management; these groups are and will be the end-users who will daily, weekly, monthly or yearly will interact with the reporting component of the IS. Since there was no CASE tools support with graphical representation techniques, we resorted to alternative Process and MetaCASE tool technology (MetaEdit+) that supported semi-formal notations in automated mode and we also initially depicted these user groups and systems requirements in the equivalent matrices of the formal diagrams (see Appendix A).

After all, the essence of a software-based paper mill information system is a construct of interlocking concepts: data sets, relationships among data/events, algorithms and invocations of functions. The essence is abstract and the conceptual design constructs need to be highly precise and richly detailed. Designing and building a testable software system to meet the end-users requirements specification and subsequent verification and validation, one should be concerned with the description of formal computational properties of the application domain (paper mill). In this sense, our proposed design solution relates two formal systems, software and hardware, which consist of a multiplicity of inter-connected components, sensors, former databases with unrefined event gatherings and so on. Now, this application domain is defined as an event log and as a formal computational system.

Acknowledgements: This research work has been done for PATEA (*PAperitehtaan TEhokkuuden Arviointijärjestelmä*) Project, and it was mainly carried out in Information Technology Research Institute of Jyväskylä University and in Tampere University of Technology. The research activities were co-ordinated by Jyväskylä Science Park and funded by the National Research and Development Agency of Finland TEKES. The authors would like to thank Prof. Risto Ritala, Tampere University of Technology, Dr Eero Haarla, Helsinki UPM-Kymmene Forest Products, and Mr Markku Suokas, Lappeenranta TietoEnator IT Services and Consultancy, who, as project partners actively participated in previous co-authoring and provided useful insights and constructive comments throughout the PATEA project work. We would also like to extend our thanks to the local paper mills production managers for the data and observations they provided us with.

6. References

1. Haarla A. (2003). Product Differentiation: Does it provide competitive advantage for a printing paper company? Ph.D. Thesis, Helsinki University of Technology, Dept. of Forest Products Technology, Laboratory of Paper Technology, Helsinki, Finland.
2. Hameri A.-P., Nikkola J. (2002). A benchmark study of 12 fine paper machines on operational efficiency. *Paper and Timber*, Vol. 84, No. 4.
3. Robinson M, Kovalainen M, Auramäki E. (2000). Diary as dialogue in paper mill process control, *Communications of the ACM*. Volume 43, Number 1, pp. 65-70.
4. Auramäki E, Robinson M, Aaltonen A, Kovalainen M, Liinamaa A & Tuuna-Väiskä T (1996). Paperwork at 78 k.p.h., *Proceedings of CSCW'96*, Boston, USA. ACM Press.
5. Berki E., Lehmonen, J., Manninen, V., Veijalainen, J., Moisanen, P., Ritala, R., Haarla, E. & Suokas, M. (2005). Requirements Analysis for Process and Product Quality Improvements: Building-in Software Quality by Designing an Information System for Assessing and Analysing Efficiency of Paper Mills. Bennetts, P., Ross, M. & Staples, J., (Eds) *Software Quality Management XIII, Current Issues in Software Quality*. The 13th International Software Quality Management Conference, (SQM 2005), 19-21 Mar 2005, Cheltenham, UK. pp. 139-165. British Computer Society.
6. Airola N & Haarla E (2004). Production Indices for Paper Production. The Association of Pulp and Paper Chemists and Engineers, Expert Committee for Paper Production. Proposal version 1.0, 28.3.2004, Zellcheming Paper Committee.

7. Veijalainen J, Berki E, Lehmonen J, Manninen V, Moisanen P, Annala J & Ritola R. (2005). PATEA PROJECT. Analyzing and assessing the efficiency of paper mills. Final report 27.07.2005. the University of Jyväskylä and the Technical University of Tampere, Finland.
8. Fuggetta A, Ghezzi C, Mandrioli D & Morzenti A (1993). Executable Specifications with Data-flow Diagrams, *Software Practice and Experience*, Vol. 23(6), 629-653
9. Berki E & Novakovic D. (1995). Towards an Integrated Specification Environment (ISE). 5th Hellenic International Conference on Informatics, Greek Computer Society, Athens.
10. Berki E & Georgiadou E (1996). Resolving Data Flow Diagramming Deficiencies by using Finite State Machines. 5th International Conference on Software Quality, Dundee.
11. France, R. (1992). Semantically extended data flow diagrams: A formal specification tool. *IEEE Transactions on Software Engineering*, Vol.18, No.4, Apr.
12. Docker, T. W. G. & Tate, G. (1986). Executable data flow diagrams. in D. Barnes and P. Brown (Eds) Proceedings of the BCS/IEE Conference 'Software Engineering 86', P. Peregrinus Ltd, pp. 352-370.
13. Berki E, Georgiadou E, Holcombe M (2004). Requirements Engineering and Process Modelling in Software Quality Management - Towards a Generic Process Metamodel. *Software Quality Journal*, 12, pp. 267-285, Kluwer Academic Publishers.
14. Davis, A.M. (1988). A Comparison of Techniques for the Specification of External System Behaviour. *Communications of the ACM*, Volume 31, No 9, Sep.
15. Olive A. (2000). Time and Change in Conceptual Modeling of Information Systems. In the Proc. of Brinkkemper, S., Lindencrona, E. & Solvberg, A. (Eds): *Information Systems Engineering, State of the Art & Research Themes*, pp. 209-220, Springer-Verlag London.
16. Berki E (2001). Establishing a scientific discipline for capturing the entropy of systems process models. CDM-FILTERS: A Computational and Dynamic Metamodel as a Flexible & Integrated Language for the Testing, Expression and Re-engineering of Systems. PhD Thesis, Faculty of Science, Computing and Engineering, University of North London, UK.
17. Lewis H.R. & Papadimitriou C.H. (1998). *Elements of the Theory of Computation*. Prentice Hall International Editions.
18. Wood D. (1987). *Theory of Computation*. J. Wiley and Sons.
19. Eilenberg, S. (1974). *Automata, Languages and Machines*. Vol. A, Academic Press.
20. Holcombe M (1988). X-Machines as a Basis for Dynamic System Specification. *Software Engineering Journal*, March.
21. Fencott, C. (1996). *Formal Methods for Concurrency*. International Thomson Computer Press.
22. Diller, A. (1990). *Z - An introduction to formal methods*. John Wiley.
23. Cooling, J. E. (1991). *Software Design for Real-Time Systems*. Chapman & Hall.
24. Ward P.T. & Mellor S.J. (1985). *Structured Development for Real-Time Systems*. Yourdon Press, New York.
25. Hatley D.J. & Pirbhai I (1987). *Strategies for Real-Time System Specification*. New York: Dorset.
26. Turksi, W.M. (1986). And no philosopher's stone, either. *Information Processing '86, Proc. IFIP, 10th World Computer Congress*, pp. 1077-1080, North Holland.
27. Liebig, C., Cilia, M. & Buchmann, A. (1999). Event Composition in Time-dependent Distributed Systems. *Proc. of IFCIS International Conference on Cooperative Information Systems*. pp. 70 - 78.
28. Hinchey M. G. & Bowen J. P. (1995). *Applications of Formal Methods*. Prentice Hall International Series in Computer Science. C.A.R. Hoare Series Editor.

APPENDIX A: Real Time States and Event transition matrix in Paper Mill (Start state on 1st column)

	Time Available (TNA) Not Available (TNA) no personnel available	TNA Personnel Available	Planned Shutdown	Unplanned Shutdown	Production Time	Start-Up Time	Grade Change	Technical /PT Work	Break
Time Not Available (TNA) no Personnel Available		holidays. work agreements, strikes	Manager Decision	Manager Decision		STOCK ON WIRE			
TNA Personnel Available	holidays. work agreements, strikes					STOCK ON WIRE			
Planned Shutdown	holidays. work agreements, strikes	Transition period in bigger rebuilds		Planned shutdown time expires		STOCK ON WIRE			
Unplanned Shutdown	holidays. work agreements, strikes	Rebuild, unexp. crashes > 48h Lack of Orders, Mat. Shortage, Unexp. events >24h		reason of shutdown changes		STOCK ON WIRE			
Production Time			(goes through Idle time)	(goes through Idle time)			Paper wound on reeler (PNWOR) + ?	PNWOR+?	PNWOR+?
Start-Up Time				Stock off wire	Paper wound on reeler PWOR				
Grade Change				Stock off wire	PWOR+ ?				
Technical / PT Work			Stock off wire + planned SD time begins	Stock off wire	PWOR+ ?				
Break				Stock off wire	(PWOR) + ?				