Timed Data Flow Diagrams

Jürgen Symanzik and Albert L. Baker

TR #96–23

December 1996

Keywords: Statistical Software Engineering, Formal Methods, Concurrent and Distributed Systems, Software Specification, Formalized Data Flow Diagrams, Timed Petri Nets.

Department of Computer Science

226 Atanasoff Hall

Iowa State University

Ames, Iowa 50011–1040, USA

1 TIMED DATA FLOW DIAGRAMS

Abstract

Data Flow Diagrams (DFD's) are widely used in industry to express requirements specifications. However, as used in practice, there has been no precise semantics for DFD's, let alone an incorporation of a model of time. In this paper, we augment the Formalized Data Flow Diagrams (FDFD's) defined in [LWBL96] by adding a deterministic (or stochastic) time behavior for the consumption of values from in–flows to processes and the production of values to the out–flows from processes. We call our new FDFD model Timed (or Stochastic) Data Flow Diagrams (TDFD's or SDFD's). We identify two factors in determining how time can affect the choice of how an FDFD can change state. The first factor has to do with when the decision is made as to which state transition will be next occur. The two possibilities are a *Preselection Policy* and a *Race Policy*. The other timing factor is the past history of an FDFD execution. We identify three alternatives: *Resampling*, *Work Age Memory*, and *Enabling Age Memory*. Certain combinations of these alternatives allow us to model systems where components are competing for limited resources. Other combinations allow us to model systems where components work concurrently. Preemption can also be modeled using these alternatives. Major results for the quantitative and qualitative analysis of TDFD's can be borrowed from the literature on Timed Petri Nets.

1.1 Introduction

If formal specification methods, at an appropriate level of abstraction, could support

- reasonable and convenient modeling of system timing behavior and
- direct specification execution

their use would be far more prevalent. The results presented in this paper provide a formalization of an already widely–used specification method that supports modeling of the timing behavior of concurrent and distributed systems and that can be directly executed. Our approach is to integrate models of timing behavior and semantic rigor with traditional Data Flow Diagrams (DFD's).

DFD's are the basis of the software development methodology known as "Structured Analysis" (SA) ([DeM78], [WM85a]). DFD's are popular because their graphical representation and hierarchical structure allow some comprehension by users with non–technical backgrounds and they can also serve as an initial characterization of software architecture.

The primary components of DFD's are bubbles and flows. In the graphical representation, bubbles are drawn as circles while flows are drawn as arcs connecting the bubbles. Hence, formally, a DFD is a directed bipartite graph. Within this model, bubbles can represent processes in a distributed or concurrent system. Flows can then represent message paths. A bubble consumes the information (values) on its in–flows, and produces information on its out–flows.

Numerous formalizations of DFD's have appeared in the technical literature, e. g., in [DeM78], [WM85a], [WM85b], [Har87], [TP89], [You89], [Har92], and [Har96]. In this paper we use the definitions of Formalized Data Flow Diagrams (FDFD's) developed by Coleman, Wahls, Baker, and Leavens in [Col91], [CB94], [WBL93], and [LWBL96]. [War86] introduces a transformation schema that allows to represent the control and timing aspects of a real system modeled as a DFD. However, this approach has very little in common with computational models such as Timed Petri Nets (TPN's) or the concept of time in FDFD's, introduced in this paper, where time is used to describe the behavior and to analyze quantitative properties of the real system. In particular, for the definitions and example in this paper, we use the notation from [LWBL96]. The potential for direct execution of these FDFD's is presented in [WBL94].

It has been shown recently that a subclass of FDFD's, so called persistent flow–free Reduced Data Flow Diagrams (PFF–RDFD's) is Turing equivalent ([SB96a]). On the other hand, features such as persistent flows, stores, and the facility to test for empty flows that are widely used in applications of

FDFD's, only add to the expressive convenience of FDFD's, and do not raise the power of the model beyond that of Turing Machines ([SB96b]).

To–date, and to the best of our knowledge, there does not exist any extension of DFD's that includes the notion of time. We have augmented FDFD's to include timing, and refer to such DFD's as Timed Data Flow Diagrams (TDFD's) or Stochastic Data Flow Diagrams (SDFD's) acknowledging the stochastic models we adopt for time. Because of the wide use of DFD's for requirements specifications, we completely maintain the original syntax and semantics of FDFD's and, in what we hope is both an intuitive and general manner, add a model of time to the operational semantics of FDFD's.

We have borrowed from the work on Timed Petri Nets (TPN's), in particular from [MBB+85], for incorporating a model of time into FDFD's. However, TPN's are used primarily to capture the requisite synchronization in concurrent and distributed systems, but do not usually represent the full functional behavior of the systems. Thus, if one is developing a client server system with replicated servers, TPN's can be used to indicate the synchronization of communication between servers in satisfying a client request, but would not capture the particulars of the data interchanged between servers, nor the actual responses to clients. By relating the existing analytical results for TPN's to our model of time in TDFD's, people who currently use DFD's as a specification technique can immediately use the more powerful timed model and achieve the same type of results for issues like deadlock and race conditions available for analogous TPN's.

A reasonable approach to the modeling of time in FDFD's is to define a stochastic time behavior for the consumption of in–flow items as well as a stochastic time behavior for the production of items on the out–flow. In a different approach, we could assign message passing times to the FDFD. Other models for times, or mixtures of several approaches, might be adopted. In this paper we only use the first approach. But it is worth noting that we have found our model of time to be expressively convenient and that the particular choice may not be of theoretical importance. It is established in [BR90] that, on a fundamental level, any type of TPN is sufficient, as long as it contains nonzero delays. We elsewhere ([SB96a], [SB96c]) argue the tight relationship between (subclasses of) FDFD's, Petri Nets (e. g., [Pet81]), and FIFO Petri Nets (introduced in [MM81]).

The work presented here covers one of the interrelated aspects of Statistics and Software Engineering, combined in the new interdisciplinary field "Statistical Software Engineering"[1]. The introduction of timing and the related statistical analysis will allow as early as in the Specifications phase of the spiral

[1] Statistical Software Engineering: "The interdisciplinary field of statistics and software engineering specializing in the use of statistical methods for controlling and improving the quality and productivity of the practices used in creating software." ([Nat96], p. 5)

software development process model ([Nat96], p. 63) to decide whether quantitative requirements of the software system are fulfilled.

In Section 1.2 of this paper, we will summarize basic definitions for FDFD's. Timed (Stochastic) Data Flow Diagrams will be introduced in Section 1.3. In Section 1.4, we describe a Producer/Consumer Model as a TDFD and consider possible execution policies. We conclude this paper with an overview on future work in Section 1.5.

1.2 Definitions

The definition of FDFD's from [LWBL96] is:

Definition (1.2.1): A *Formalized Data Flow Diagram* (FDFD) is a quintuple

$$FDFD = (B, FLOWNAMES, TYPES, P, F),$$

where B is a set of *bubbles*, $FLOWNAMES$ is a set of *flows*, $TYPES$ is a set of *types*, P is the set $\{persistent, consumable\}$ and $F = B \times FLOWNAMES \times TYPES \times B \times P$. The following notational convention for members from these domains is used: $b \in B, fn \in FLOWNAMES, T \in TYPES, p \in P, f \in F$. ■

While a more rigorous definition of the mappings used to define the execution behavior of FDFD's is contained in [LWBL96], a less formal explanation follows.

Consumable flows are modeled as infinite queues of values and persistent flows are modeled as shared variables for which the source bubble can write the value and the destination bubble can read the value. Intuitively, the semantics of FDFD's is based on a two-step firing rule for bubbles. Each bubble b is either in one of two modes: *idle* or *working*. The state of an FDFD is the current value on all the flows and the current mode of each bubble (along with what values were consumed by bubbles in the *working* mode, at the point they went from *idle* to *working*).

If bubble b is *idle*, then its change of state to *working* is predicated by an enabling rule, *Enabled* which is just an assertion over the values on its in–flows. If *Enabled* is satisfied for bubble b and the values of the in–flows to b, then b is a candidate for firing. If b is selected for firing, then the values on the in–flows to b which satisfied the enabling rule are consumed (or copied from persistent flows), and b enters *working* mode.

If a bubble b is *working*, it is also a candidate for firing. When b is selected for execution, the values that were previously consumed (or copied) are used in a postcondition assertion that defines the values to be output from b on out–flows from b.

From this basic definition of FDFD's, we can define a sequence of firing steps in the execution of an FDFD:

Definition (1.2.2): A *firing sequence* (*computation sequence*) of an FDFD is a possibly infinite sequence $(b_i, a_i, j_i) \in B \times \{C, P\} \times I\!N, i \geq 0$, such that, if transition (b_i, a_i, j_i) is fired in state (bm, r, fs), then

$$(fs', r') = \begin{cases} (Consume(b_i))_{j_i}(fs, r), & \text{if } a_i = C \\ (Produce(b_i))_{j_i}(fs, r), & \text{if } a_i = P \end{cases}$$

$$bm'(b_i) = \begin{cases} working, & \text{if } a_i = C \\ idle, & \text{if } a_i = P \end{cases}$$

$$bm'(b) = bm(b) \quad \forall b \in B - \{b_i\}$$

and

$$(bm, r, fs) \rightarrow (bm', r', fs').$$

We introduce the notation $(bm, r, fs)[(b, a, j)]$ to indicate that transition (b, a, j) is fireable in state (bm, r, fs) and $(bm, r, fs)[(b, a, j)](bm', r', fs')$ to indicate that state (bm', r', fs') is reached upon the firing of transition (b, a, j) in state (bm, r, fs).

By induction, we extend this notation for firing sequences:

$$(bm_0, r_0, fs_0)[(b_1, a_1, j_1), \ldots, (b_{n-1}, a_{n-1}, j_{n-1}), (b_n, a_n, j_n)]$$

is used to indicate that transition (b_n, a_n, j_n) is fireable in state $(bm_{n-1}, r_{n-1}, fs_{n-1})$, given that

$$(bm_0, r_0, fs_0)[(b_1, a_1, j_1), \ldots, (b_{n-1}, a_{n-1}, j_{n-1})](bm_{n-1}, r_{n-1}, fs_{n-1})$$

holds. By analogy, we use

$$(bm_0, r_0, fs_0)[(b_1, a_1, j_1), \ldots, (b_n, a_n, j_n)](bm_n, r_n, fs_n)$$

to indicate that state (bm_n, r_n, fs_n) is reached upon the firing of the sequence $(b_1, a_1, j_1), \ldots, (b_n, a_n, j_n)$.

■

Definition (1.2.3): The *set of firing sequences* (*set of computation sequences, language*) of an FDFD, denoted by $FS(FDFD, \gamma_{initial})$, is the set containing all firing sequences that are possible for

this FDFD, given $\gamma_{initial} = (bm_{initial}, r_{initial}, fs_{initial})$, i. e.,

$$FS(FDFD, \gamma_{initial}) = \{s \mid s \in (B \times \{C, P\} \times I\!N)^* \wedge \gamma_{initial}[s]\}. \quad \blacksquare$$

Definition (1.2.4): The *Reachability Set* of an FDFD, denoted by $RS(FDFD, \gamma_{initial})$, is the set of states $\gamma = (bm, r, fs)$ that are reachable from $\gamma_{initial} = (bm_{initial}, r_{initial}, fs_{initial})$, i. e.,

$$RS(FDFD, \gamma_{initial}) = \{\gamma \mid \gamma \in , \ \wedge \exists\, s \in FS(FDFD, \gamma_{initial}) \ : \ \gamma_{initial}[s]\gamma\}. \quad \blacksquare$$

Definition (1.2.5): Let $EN(\gamma) \subseteq B \times \{C, P\} \times I\!N$ be the set of transitions that are enabled in state $\gamma = (bm, r, fs)$, i. e.,

$$EN(\gamma) = \{s \mid s \in (B \times \{C, P\} \times I\!N) \wedge \gamma[s]\}. \quad \blacksquare$$

1.3 Stochastic Data Flow Diagrams

We will make use of random variables to specify the time behavior of FDFD's. Therefore, Stochastic Data Flow Diagram (SDFD) is a more appropriate name for our new model. We try to follow the general approach of Stochastic Petri Nets given in [MBB+85] when defining SDFD's, when considering the impact of different execution policies on the semantics of the model, and when allowing a general time distribution that induces an associated stochastic process.

We start to describe the behavior of a TDFD by describing a possible timed firing sequence of an FDFD.

Definition (1.3.1): A *timed firing sequence* (TFS) of an FDFD with initial state $\gamma_{initial}$ is a pair $tfs = (s, \tau)$, where $s \in FS(FDFD, \gamma_{initial})$ and τ is a non–decreasing sequence (of the same length) of real non–negative values representing the instants of firing (called *epochs*) of each transition, such that consecutive transitions (b_i, a_i, j_i) and $(b_{i+1}, a_{i+1}, j_{i+1})$ correspond to ordered epochs $\tau_i \leq \tau_{i+1}$. The time intervals $[\tau_i, \tau_{i+1})$ between consecutive epochs represent the periods in which the FDFD remains in state γ_i (assuming $\tau_0 = 0$). A *history* of the FDFD up to the kth epoch τ_k is denoted by $Z(k)$. $\blacksquare$

The introduction of a stochastic time behavior for FDFD's will allow us to describe (in a probabilistic sense) the future behavior of a system from the knowledge of the past history and the current state.

Definition (1.3.2): Let $\underline{Z} = Z(k)$ be a history of the FDFD up to (and including) the kth epoch, and $\underline{\gamma} = \gamma_k$ be the state entered by firing transition (b_k, a_k, j_k). We assume that for all k, $\underline{Z}$, and $\underline{\gamma}$, the following joint distribution functions can be uniquely determined:

$$F_{\Gamma,X}((b,a,j),x \mid \underline{\gamma},\underline{Z}) = Pr(\Gamma = (b,a,j), X \leq x \mid \underline{\gamma},\underline{Z}) \qquad (1.3.2.1)$$

■

The distribution above depends on two random variables: The discrete random variable Γ represents the transition that will fire. The sample space for Γ is the set of transitions enabled in $\underline{\gamma}$, i. e., $\Omega_\Gamma = EN(\underline{\gamma})$. The continuous random variable X represents the time that elapses from entering $\underline{\gamma}$ up to the next transition epoch, i. e., the time interval $\tau_{k+1} - \tau_k$. The sample space for X are positive real numbers including 0 (same time as previous transition) and ∞ (never), i. e., $\Omega_X = I\!R^+_\infty$.

Note that $\underline{\gamma}$ is known from $\underline{Z}$, but we have explicitly indicated the dependence on $\underline{\gamma}$ since quite often, $\underline{\gamma}$ is the only component that influences the joint distribution functions (1.3.2.1). These distributions must be defined for all transitions $(b,a,j) \in EN(\underline{\gamma})$.

We define the marginal probability function of selecting (b,a,j) to be the next transition to fire as

$$\begin{aligned} p_{(b,a,j)}(\underline{\gamma},\underline{Z}) &= Pr(\Gamma = (b,a,j) \mid \underline{\gamma},\underline{Z}) \\ &= \lim_{t\to\infty} F_{\Gamma,X}((b,a,j),t \mid \underline{\gamma},\underline{Z}) \\ &= \int_0^\infty d_x F_{\Gamma,X}((b,a,j),x \mid \underline{\gamma},\underline{Z}) \qquad (1.3.2.2) \end{aligned}$$

and the marginal distribution function of the time spent in state $\underline{\gamma}$ before the next epoch is reached as

$$F_X(x \mid \underline{\gamma},\underline{Z}) = \sum_{s\in EN(\underline{\gamma})} F_{\Gamma,X}(s,x \mid \underline{\gamma},\underline{Z}). \qquad (1.3.2.3)$$

Definition (1.3.3): A *Stochastic Data Flow Diagram* (SDFD) is an FDFD (with initial state $\gamma_{initial}$) with a set of specifications for calculating the joint distribution functions $F_{\Gamma,X}((b,a,j),x \mid \underline{\gamma},\underline{Z})$ for all $\underline{\gamma}$ and $\underline{Z}$, and with an initial probability distribution on the Reachability Set $RS(FDFD,\gamma_{initial})$.

■

Due to this definition, the ensemble of possible executions of a SDFD, together with the probability measure induced on it by assigning the firing distributions $F_{\Gamma,X}((b,a,j),x \mid \underline{\gamma},\underline{Z})$, describes a stochastic process with a discrete state space isomorphic to a subset of $RS(FDFD,\gamma_{initial})$ of the associated FDFD. For the initial probability distribution on $RS(FDFD,\gamma_{initial})$, we assume that $Pr(\text{system is in state } \gamma_{initial} \text{ at time } \tau_0 = 0) = 1$.

When in a given state $\underline{\gamma}$ only one transition is enabled, say (b, a, j), the calculation of

$$F_{\Gamma,X}((b,a,j),x \mid \underline{\gamma},\underline{Z})$$

requires only to determine the distribution of the time spent in $\underline{\gamma}$, possibly conditioned on the past history $\underline{Z}$.

When a state $\underline{\gamma}$ enables at least two transitions, the computation of the distributions

$$F_{\Gamma,X}((b,a,j),x \mid \underline{\gamma},\underline{Z})$$

requires the knowledge of the policy used for the selection of the transition that fires. We consider two possible policies:

Definition (1.3.4): The *Preselection Policy*: When the SDFD enters state $\underline{\gamma}$, a transition s is selected among those in $EN(\underline{\gamma})$ according to its probability $p_s(\underline{\gamma},\underline{Z})$, Then, s will fire after a random delay with distribution $F_{X|\Gamma}(x \mid s,\underline{\gamma},\underline{Z})$. Thus, the selection of the transition that actually fires does not depend upon the associated delay. Otherwise, once a transition has been selected, the sojurn time in $\underline{\gamma}$ does not depend upon the delays associated to the other transitions. Therefore, a model with preselection policy requires the specification of both the probabilities $p_s(\underline{\gamma},\underline{Z})$ and the conditional distributions $F_{X|\Gamma}(x \mid s,\underline{\gamma},\underline{Z})$.

Equation (1.3.2.1) can be rewritten as

$$F_{\Gamma,X}(s,x \mid \underline{\gamma},\underline{Z}) = p_s(\underline{\gamma},\underline{Z}) \cdot F_{X|\Gamma}(x \mid s,\underline{\gamma},\underline{Z}) \qquad (1.3.4.1)$$

where $F_{X|\Gamma}$ represents the conditional distribution of the firing delays conditioned on $\underline{\gamma},\underline{Z}$, and the fact that s is the transition that will actually fire. Obviously, $\sum_{s_i \in EN(\underline{\gamma})} p_{s_i}(\underline{\gamma},\underline{Z}) = 1$. ■

Definition (1.3.5): The *Race Policy*: When the SDFD enters state $\underline{\gamma}$, for each transition $s_i \in EN(\underline{\gamma})$ a random sample ϑ_i from θ_i is extracted from the joint distribution

$$\phi_{\theta_1,\ldots,\theta_{|EN(\underline{\gamma})|}}(x_1,\ldots,x_{|EN(\underline{\gamma})|} \mid \underline{\gamma},\underline{Z}) = Pr(\theta_1 \le x_1,\ldots,\theta_{|EN(\underline{\gamma})|} \le x_{|EN(\underline{\gamma})|} \mid \underline{\gamma},\underline{Z}) \qquad (1.3.5.1)$$

The minimum ϑ_i of the samples $\vartheta_1,\ldots,\vartheta_{|EN(\underline{\gamma})|}$ determines two properties: the transition s_i which will actually fire and the sojorn time ϑ_i in $\underline{\gamma}$. We consider only the case where all random variables θ_i are stochastically independent. Then, (1.3.5.1) is uniquely determined by the marginal distributions

$$\phi_i(x \mid \underline{\gamma},\underline{Z}) = Pr(\theta_i \le x \mid \underline{\gamma},\underline{Z}), i = 1,\ldots,\mid EN(\underline{\gamma}) \mid. \qquad (1.3.5.2)$$

Now, the joint distribution function (1.3.2.1) can be expressed as

$$F_{\Gamma,X}(s_i, x \mid \underline{\gamma}, \underline{Z}) = \int_0^x \left(\prod_{\substack{j=1,\ldots,|EN(\underline{\gamma})| \\ j \neq i}} [1 - \phi_j(u \mid \underline{\gamma}, \underline{Z})] \right) d_u \phi_i(u \mid \underline{\gamma}, \underline{Z}). \qquad (1.3.5.3)$$

Therefore, the marginal probability function (1.3.2.2) can be rewritten as

$$\begin{aligned} p_{s_i}(\underline{\gamma}, \underline{Z}) &= \lim_{t \to \infty} F_{\Gamma,X}(s_i, t \mid \underline{\gamma}, \underline{Z}) \\ &= \int_0^\infty \left(\prod_{\substack{j=1,\ldots,|EN(\underline{\gamma})| \\ j \neq i}} [1 - \phi_j(u \mid \underline{\gamma}, \underline{Z})] \right) d_u \phi_i(u \mid \underline{\gamma}, \underline{Z}) \qquad (1.3.5.4) \end{aligned}$$

and the marginal distribution function (1.3.2.3) can be expressed as

$$\begin{aligned} F_X(x \mid \underline{\gamma}, \underline{Z}) &= 1 - \prod_{j=1,\ldots,|EN(\underline{\gamma})|} Pr(\theta_j > x \mid \underline{\gamma}, \underline{Z}) \\ &= 1 - \prod_{j=1,\ldots,|EN(\underline{\gamma})|} [1 - \phi_j(x \mid \underline{\gamma}, \underline{Z})]. \qquad (1.3.5.5) \end{aligned}$$

Thus, a model with race policy requires the specification of the marginal distributions $\phi_i(x \mid \underline{\gamma}, \underline{Z}), i = 1, \ldots, \mid EN(\underline{\gamma}) \mid$, only. ■

Now, we consider three possible ways to deal with the past history $\underline{Z}$:

Resampling: The distributions $F_{\Gamma,X}((b,a,j), x \mid \underline{\gamma}, \underline{Z})$ are independent of $\underline{Z}$, but they may depend on the current state $\underline{\gamma}$.

Work Age Memory: The distributions $F_{\Gamma,X}((b,a,j), x \mid \underline{\gamma}, \underline{Z})$ depend on the past history $\underline{Z}$ through a new variable, a so–called work age variable, associated with each transition (b,a,j). The work age variables accumulate the work done — for each transition from its last firing up to the considered epoch. The distributions $F_{\Gamma,X}((b,a,j), x \mid \underline{\gamma}, \underline{Z})$ represent the residual times needed for the transitions to complete.

Enabling Age Memory: The distributions $F_{\Gamma,X}((b,a,j), x \mid \underline{\gamma}, \underline{Z})$ depend on the past history $\underline{Z}$ through a new variable, a so–called enabling age variable, associated with each transition (b,a,j). The enabling age variables accumulate the work done — for each transition from the last instant at which it has become enabled up to the considered epoch. The distributions $F_{\Gamma,X}((b,a,j), x \mid \underline{\gamma}, \underline{Z})$ represent the residual times needed for the transitions to complete.

Combining the two policies for selecting a transition that fires with the three methods of dealing with the past history $\underline{Z}$ results in six different execution policies. Four of them are intuitively easy to understand while two of them are more complex and can be neglected for practical purposes.

Race Policy with Resampling (RR): This execution policy best describes the behavior of a set of parallel competing (conflicting) transitions. The first transition to terminate determinates a change in the system state. The work done by all other transitions that do not complete is lost. Therefore, this policy seems to be interesting only in the case of conflicting transitions that make use of the same resources. Note that this only happens for FDFD's if for a given input that has been read a bubble can nondeterministically select among two or more alternatives which output to produce when changing its mode from *working* to *idle*.

Race Policy with Work Age Memory (RW): Here, simultaneous transitions are described. The first transition to terminate determines a change in the system state. However, the work done by all other transitions that do not complete is not lost. Instead, it is assumed that all the work done by each transition is being accumulated from when it is first enabled up to the current firing. After this firing, a transition will resume its work in the next state that enables it, from the point at which it has been interrupted. Therefore, this execution policy is useful in situations where conflicting and concurrent transitions can happen.

Race Policy with Enabling Age Memory (RE): Once again, simultaneous transitions are described. The first transition to terminate determines a change in the system state. However, in this case, the work done by all other transitions that do not complete is lost, unless they remain enabled in the new state that is reached through the current firing. Therefore, this execution policy is useful in situations where conflicting and concurrent transitions can happen, but it also allows preemption.

Preselection with Resampling (PR): This execution policy can be used when a set of conflicting transitions can not perform in parallel. Before the system can enter a new state, it has to choose which of the conflicting transitions will fire next. Once choosen, this transition will perform until completion and the system will enter the new state.

Preselection with Work Age Memory (PW): In this execution policy, the transition that will determine the stage change is choosen immediately when a new state has been reached. Then, this transition performs until completion and the system will enter the next state. All other transitions that were enabled but have not been choosen execute in parallel to the choosen one until the state change caused by the choosen transition occurs. Each transition will resume work in the next state where it is enabled, continuing from the point that has been reached when the state change occurred. Of course, this may cause some paradoxon: It may happen that the

choosen transition performs longer than some of the transitions that have not been choosen and have terminated without being allowed to induce a state change.

Preselection with Enabling Age Memory (PE): Similarly, the transition that will determine the stage change is choosen immediately when a new state has been reached. Then, this transition performs until completion and the system will enter the next state. All other transitions that were enabled but have not been choosen execute in parallel to the choosen one until the state change caused by the choosen transition occurs. However, if a transition is not enabled in the new state that is reached through the current firing, all its accumulated work is lost. The same paradoxon as in the previous case may occur.

1.4 Example of a Producer/Consumer Model

The example in this section shows a Producer/Consumer model where the producer can work whenever it is ready, while the consumer has to wait for inputs from the producer. The given distributions do not really provide a useful application, but have been choosen to demonstrate the different behavior of the model under different execution policies. In fact, for most of the policies, the producer will produce items at a faster rate then the consumer can consume, resulting in a (permanently) increasing queue length of flow f.

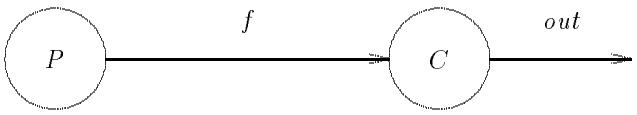

Figure 1.1: Example of a SDFD.

The mappings *Enabled*, *Consume*, and *Produce* for the SDFD shown in Figure 1.1 are specified as follows:

$$Enabled(P) = \lambda fs\ .\ true$$

$$Enabled(C) = \lambda fs\ .\ (\neg IsEmpty(f) \wedge Head(fs(f)) = 0)$$
$$\vee (\neg IsEmpty(f) \wedge Head(fs(f)) = 1)$$

	s	t_s
$Consume(P) = \lambda(fs, r)\ .\ \{(fs, r)\}$	(P, C, 1)	2
$Consume(C) = \lambda(fs, r)\ .$		

$\{$**if** $(\neg IsEmpty(f) \wedge Head(fs(f)) = 0)$

then $In(f, C)(fs, r)$ (C, C, 1) 4

fi,

if $(\neg IsEmpty(f) \wedge Head(fs(f)) = 1)$

then $In(f, C)(fs, r)$ (C, C, 2) 5

fi

$\}$

$Produce(P) = \lambda(fs, r)$.

$\{ Out(0, f, P)(fs, r),$ (P, P, 1) 2

$Out(1, f, P)(fs, r)\}$ (P, P, 2) 3

$Produce(C) = \lambda(fs, r)$.

$\{$**if** $r(C)(f) = 0$

then $Out(a, out, C)(fs, r)$ (C, P, 1) 3

$\square\, Out(b, out, C)(fs, r)$ (C, P, 2) 4

fi,

if $r(C)(f) = 1$

then $Out(c, out, C)(fs, r)$ (C, P, 3) 6

fi

$\}$

Now, we consider the effect of different execution policies on this basic model. The symbol $\surd$ marks the transition that actually fires. With "Acc" we denote the time accumulated for a transition that did not fire.

Race Policy:

We define the following marginal distributions $\phi_i(x \mid \underline{\gamma}, \underline{Z})$ for the race policy:

$$Pr(\theta_s = t_s \mid \underline{\gamma}, \underline{Z}) = 1 \ \forall s \in EN(\underline{\gamma}) \ \forall \underline{\gamma} \ \forall \underline{Z}$$

All these distributions are Dirac distributions. From the statistical point of view, they are degenerate distributions where all the mass is assigned to the point t_s. From the applied point of view, these distributions are used to express a deterministic time for each transition.

Race/Resampling

There is only one possible timed firing sequence $tfs = (s, \tau)$ for this execution policy:

i	0	1	2	3	4
τ	0	2	4	6	8
s		(P, C, 1)	(P, P, 1)	(P, C, 1)	(P, P, 1)
$EN(\underline{\gamma})$	(P, C, 1)$\surd$	(P, P, 1)$\surd$	(P, C, 1)$\surd$	(P, P, 1)$\surd$	(P, C, 1)
		(P, P, 2)	(C, C, 1)	(P, P, 2)	(C, C, 1)
				(C, C, 1)	

- $\tau_1 = 2$: (P, C, 1) is the only possible transition and executes at time 2.
- $\tau_2 = 4$: (P, P, 1) and (P, P, 2) compete and will terminate at time 4 and 5, respectively. Therefore, (P, P, 1) executes at time 4.
- $\tau_3 = 6$: (P, C, 1) and (C, C, 1) compete and will terminate at time 6 and 8, respectively. Therefore, (P, C, 1) executes at time 6.
- $\tau_4 = 8$: (P, P, 1), (P, P, 2), and (C, C, 1) compete and will terminate at time 8, 9, and 10, respectively. Therefore, (P, P, 1) executes at time 8.
- $\tau_5 = 10$: In analogy to τ_3.

Race/Work Age

We indicate the timed firing sequence $tfs = (s, \tau)$ for the first 5 epochs:

i	0	1	2	3	4	5
τ	0	2	4	6	7	8
s		(P, C, 1)	(P, P, 1)	(P, C, 1)	(P, P, 2)	(C, C, 1)
$EN(\underline{\gamma})$/Acc	(P, C, 1)/0 $\surd$	(P, P, 1)/0 $\surd$	(P, C, 1)/0 $\surd$	(P, P, 1)/0	(P, C, 1)/0	(P, C, 1)/1
		(P, P, 2)/0	(C, C, 1)/0	(P, P, 2)/2 $\surd$	(C, C, 1)/3 $\surd$	(C, P, 1)/0
				(C, C, 1)/2		(C, P, 2)/0
$\neg EN(\underline{\gamma})$/Acc			(P, P, 2)/2		(P, P, 1)/1	(P, P, 1)/1

- $\tau_1 = 2$: (P, C, 1) is the only possible transition and executes at time 2.
- $\tau_2 = 4$: (P, P, 1) and (P, P, 2) compete and will terminate at time 4 and 5, respectively. Therefore, (P, P, 1) executes at time 4. (P, P, 2) accumulates 2 time units of work, but is not enabled past $\tau_2 = 4$.
- $\tau_3 = 6$: (P, C, 1) and (C, C, 1) compete and will terminate at time 6 and 8, respectively. Therefore, (P, C, 1) executes at time 6. (C, C, 1) accumulates 2 time units of work and is enabled past $\tau_3 = 6$. (P, P, 2) is enabled again past $\tau_3 = 6$.

- $\tau_4 = 7$: (P, P, 1), (P, P, 2), and (C, C, 1) compete and will terminate at time 8, 7, and 8, respectively. Therefore, (P, P, 2) executes at time 7. (P, P, 1) accumulates 1 time unit of work, but is not enabled past $\tau_4 = 7$. (C, C, 1) accumulates 1 time unit of work and is enabled past $\tau_4 = 7$.

- $\tau_5 = 8$: (P, C, 1) and (C, C, 1) compete and will terminate at time 9 and 8, respectively. Therefore, (C, C, 1) executes at time 8. (P, C, 1) accumulates 1 time unit of work and is enabled past $\tau_5 = 8$. (P, P, 1) remains disabled past $\tau_5 = 8$.

Race/Enabling Age

We indicate the timed firing sequence $tfs = (s, \tau)$ for the first 5 epochs:

i	0	1	2	3	4	5
τ	0	2	4	6	8	10
s		(P, C, 1)	(P, P, 1)	(P, C, 1)	(P, P, 1)	(P, C, 1)
					(C, C, 1)	
$EN(\underline{\gamma})$/Acc	(P, C, 1)/0 $\surd$	(P, P, 1)/0 $\surd$	(P, C, 1)/0 $\surd$	(P, P, 1)/0 $\surd$	(P, C, 1)/0 $\surd$	(P, P, 1)/0
		(P, P, 2)/0	(C, C, 1)/0	(P, P, 2)/0	(C, P, 1)/0	(P, P, 2)/0
				(C, C, 1)/2 $\surd$	(C, P, 2)/0	(C, P, 1)/2
						(C, P, 2)/2

- $\tau_1 = 2$: (P, C, 1) is the only possible transition and executes at time 2.

- $\tau_2 = 4$: (P, P, 1) and (P, P, 2) compete and will terminate at time 4 and 5, respectively. Therefore, (P, P, 1) executes at time 4. (P, P, 2) looses the work accumulated so far since it is not enabled past $\tau_2 = 4$.

- $\tau_3 = 6$: (P, C, 1) and (C, C, 1) compete and will terminate at time 6 and 8, respectively. Therefore, (P, C, 1) executes at time 6. (C, C, 1) accumulates 2 time units of work and is enabled past $\tau_3 = 6$.

- $\tau_4 = 8$: (P, P, 1), (P, P, 2), and (C, C, 1) compete and will terminate at time 8, 9, and 8, respectively. We have to consider two cases:

  - (P, P, 1) executes at time 8, then (C, C, 1) executes at time 8.
  - (C, C, 1) executes at time 8, then (P, P, 1) executes at time 8.

  Since both transitions depend on different resources and the execution of one of them does not disable the other one, the results are the same. (P, P, 2) looses the work accumulated so far since it is not enabled past $\tau_4 = 8$.

- $\tau_5 = 10$: (P, C, 1), (C, P, 1), and (C, P, 2) compete and will terminate at time 10, 11, and 12, respectively. Therefore, (P, C, 1) executes at time 10. (C, P, 1) accumulates 2 time units of work and is enabled past $\tau_5 = 10$. (C, P, 2) accumulates 2 time units of work and is enabled past $\tau_5 = 10$.

Preselection Policy:

We consider three different cases of the Preselection/Resampling policy.

- First, we define

$$p_s(\underline{\gamma}, \underline{Z}) = \frac{1}{\mid EN(\underline{\gamma}) \mid} \ \forall s \in EN(\underline{\gamma}) \ \forall \underline{\gamma} \ \forall \underline{Z}$$

and

$$Pr(X = t_s \mid s, \underline{\gamma}, \underline{Z}) = 1 \ \forall s \in EN(\underline{\gamma}) \ \forall \underline{\gamma} \ \forall \underline{Z}.$$

Here are all possible timed firing sequences $tfs = (s, \tau)$ according to this policy and distribution for the first 4 epochs:

i	0	1	2	3	4
s : τ		2 : (P, C, 1)	4 : (P, P, 1)	6 : (P, C, 1)	8 : (P, P, 1)
					9 : (P, P, 2)
					10 : (C, C, 1)
				8 : (C, C, 1)	10 : (P, C, 1)
					11 : (C, P, 1)
					12 : (C, P, 2)
			5 : (P, P, 2)	7 : (P, C, 1)	9 : (P, P, 1)
					10 : (P, P, 2)
					12 : (C, C, 2)
				10 : (C, C, 2)	12 : (P, C, 1)
					16 : (C, P, 3)

- Now, we change the selection probabilities to

$$p_{(P,C,1)}(\underline{\gamma}, \underline{Z}) = 1 \ \forall \underline{\gamma} : (idle(P)) \ \forall \underline{Z}$$

$$p_{(P,P,1)}(\underline{\gamma}, \underline{Z}) = 1 \ \forall \underline{\gamma} : (working(P)) \ \forall \underline{Z}$$

and

$$p_s(\underline{\gamma}, \underline{Z}) = 0 \ \text{for all other } s \in EN(\underline{\gamma}) \ \forall \underline{Z}$$

but maintain the delay

$$Pr(X = t_s \mid s, \underline{\gamma}, \underline{Z}) = 1 \ \ \forall s \in EN(\underline{\gamma}) \ \ \forall \underline{\gamma} \ \ \forall \underline{Z}.$$

We make use of 0–probabilities to disable some transitions that are possible firing candidates according to $EN(\underline{\gamma})$. Actually, we only allow bubble P to proceed. This yields exactly the same timed firing sequence as for the Race/Resampling policy.

- Finally, we consider a more realistic model by choosing the following selection probabilities

$$p_{(P,C,1)}(\underline{\gamma}, \underline{Z}) = 1 \ \ \forall \underline{\gamma} : (idle(P) \wedge idle(C) \wedge IsEmpty(f)) \ \ \forall \underline{Z}$$

$$p_{(P,P,1)}(\underline{\gamma}, \underline{Z}) = 1 \ \ \forall \underline{\gamma} : (working(P)) \ \ \forall \underline{Z}$$

$$p_{(C,C,1)}(\underline{\gamma}, \underline{Z}) = 1 \ \ \forall \underline{\gamma} : (idle(P) \wedge idle(C) \wedge \neg IsEmpty(f)) \ \ \forall \underline{Z}$$

$$p_{(C,P,1)}(\underline{\gamma}, \underline{Z}) = 1 \ \ \forall \underline{\gamma} : (working(C)) \ \ \forall \underline{Z}$$

and

$$p_s(\underline{\gamma}, \underline{Z}) = 0 \ \text{ for all other } s \in EN(\underline{\gamma}) \ \ \forall \underline{Z}$$

but maintain the delay

$$Pr(X = t_s \mid s, \underline{\gamma}, \underline{Z}) = 1 \ \ \forall s \in EN(\underline{\gamma}) \ \ \forall \underline{\gamma} \ \ \forall \underline{Z}.$$

Once again, we make use of 0–probabilities to disable some transitions. The effect of this policy and distribution is a timed firing sequence where (P, C, 1), (P, P, 1), (C, C, 1), and (C, P, 1) alternate in this order.

1.5 Future Directions

One of the first things to be done in the future is an overview of the types of stochastic processes associated to our TDFD, similar to the characterization of the stochastic process that is underlying a Stochastic Petri Net ([CGL94]). Different types of probability distributions and execution policies will result in stochastic processes of different flavors, some of them easy to analyze and some of them difficult to capture. For Timed Petri Nets (TPN's), most work has been done for Exponential distributions (e. g., [MC87]), associated to a Markov process which is usually easy to analyze. Reasonable analytical results also can be gained for Phase–Type distributions. Other time behavior that can be found in the literature for TPN's, e. g., Deterministic timing (e. g., [MC87]), mixture of Deterministic and Exponential timing, and interval timing, should result in reasonable results for TDFD's, too.

Many problems that occur during the analysis of TDFD's have been known for a long time when analyzing TPN's. Some of these problems concern the state explosion and undecidability. However, there exists a large number of automated tools that help evaluate, analyze, and solve TPN's. To mention only a few, in [Chi85] a software package is introduced that allows the steady state and transient analysis of generalized Stochastic Petri Nets. [Cum85] describes a software package for the analysis of Stochastic Petri Nets models where transition firing times are distributed as Phase–Type. [Men85] provides a tool for the analysis of TPN's where firing only occurs within the limits of time defined by the interval $[a, b], b \geq a$. In [GM95], TimeNET, a tool especially designed for non–Markovian Stochastic Petri Nets is presented and a comparison with other Petri Net tools is given. It should be possible to reuse and extend methods, algorithms, and tools known from TPN's for TDFD's. Once adapted, one might hopefully automatically evaluate, analyze, and solve these TDFD's. Depending on the types of distributions that are allowed for the firing times, one might consider to provide software that is capable of doing an analytical analysis if a Markov chain or a (semi) Markov process is associated with the TDFD. Or, one might do simulations if everything else fails.

In addition to a quantitative analysis (performance, throughput, average load of a bubble, etc.), mostly affected by the choice of the probability distributions, TDFD's also invite a qualitative analysis (deadlock, reachability, termination, finiteness, liveness, etc.). Some answers to qualitative questions may be gained from the related FDFD (e. g., there is no deadlock state for the TDFD if the FDFD has no deadlock state), but others are not immediately achievable (it is not guaranteed that the TDFD can reach a particular state even though it can be reached for the FDFD). Future research should be directed towards the question how decidability problems for TDFD's and FDFD's are related, similar to the discussion in [God82] where liveness properties of Petri Nets and Timed Petri Nets are compared.

Acknowledgements

Symanzik's research was partially supported by a German "DAAD–Doktorandenstipendium aus Mitteln des zweiten Hochschulsonderprogramms". The authors wish to thank Herbert T. David and Kenneth J. Koehler for many valuable suggestions on the presentation of this topic.

Bibliography

[BR90] I.I. Bestuzheva and V.V. Rudnev. Timed Petri Nets: Classification and Comparative Analysis. *Automation and Remote Control, Pt. 1*, 51(10):1303–1318, 1990.

[CB94] D.L. Coleman and A.L. Baker. Synthesizing Structured Analysis and Object–Oriented Specifications. Technical Report 94-04, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, March 1994. Available by anonymous ftp from ftp.cs.iastate.edu or by e-mail from almanac@cs.iastate.edu.

[CGL94] G. Ciardo, R. German, and C. Lindemann. A Charcterization of the Stochastic Process Underlying a Stochastic Petri Net. *IEEE Transactions on Software Engineering*, 20(7):506–515, 1994.

[Chi85] G. Chiola. A Software Package for the Analysis of Generalized Stochastic Petri Net Models. In *International Workshop on Timed Petri Nets, Torino, Italy, July 1985*, pages 136–143, 1985.

[Col91] D.L. Coleman. *Formalized Structured Analysis Specifications.* PhD Thesis, Iowa State University, Ames, Iowa, 50011, 1991.

[Cum85] A. Cumani. ESP — A Package for the Evaluation of Stochastic Petri Nets with Phase–Type Distributed Transition Times. In *International Workshop on Timed Petri Nets, Torino, Italy, July 1985*, pages 144–151, 1985.

[DeM78] T. DeMarco. *Structured Analysis and System Specification.* Yourdon, Inc., New York, New York, 1978.

[GM95] R. German and J. Mitzlaff. Transient Analysis of Deterministic and Stochastic Petri Nets with TimeNET. In H. Beilner and F. Bause, editors, *Lecture Notes in Computer Science Vol. 977: Quantitative Evaluation of Computing and Communication Systems*, pages 209–223, Springer–Verlag, Berlin, Heidelberg, 1995.

[God82] H.P. Godbersen. On the Problem of Time in Nets. In C. Girault and W. Reisig, editors, *Informatik–Fachberichte Vol. 52: Application and Theory of Petri Nets*, pages 23–30, Springer–Verlag, Berlin, Heidelberg, 1982.

[Har87] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[Har92] D. Harel. Biting the Silver Bullet. *Computer*, 21(1):8–20, January 1992.

[Har96] D. Harel. Executable Object Modeling with Statecharts. In *Proceedings of the 18th International Conference on Software Engineering*, pages 246–257. IEEE Computer Society Press, January 1996.

[LWBL96] G.T. Leavens, T. Wahls, A.L. Baker, and K. Lyle. An Operational Semantics of Firing Rules for Structured Analysis Style Data Flow Diagrams. Technical Report 93–28d, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, December 1993, revised, July 1996. Available by anonymous ftp from ftp.cs.iastate.edu or by e–mail from almanac@cs.iastate.edu.

[MBB+85] M.A. Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. On Petri Nets with Stochastic Timing. In *International Workshop on Timed Petri Nets, Torino, Italy, July 1985*, pages 80–87, 1985.

[MC87] M.A. Marsan and G. Chiola. On Petri Nets with Determinstic and Exponentially Distributed Firing Times. In G. Rozenberg, editor, *Lecture Notes in Computer Science Vol. 266: Advances in Petri Nets 1987*, pages 132–145, Springer–Verlag, Berlin, Heidelberg, 1987.

[Men85] M. Menasche. PAREDE: An Automated Tool for the Analysis of Time(d) Petri Nets. In *International Workshop on Timed Petri Nets, Torino, Italy, July 1985*, pages 162–169, 1985.

[MM81] R. Martin and G. Memmi. Specification and Validation of Sequential Processes Communicating by FIFO Channels. *I.E.E. Conference Publication No. 198: Fourth International Conference on Software Engineering for Telecommunication Switching Systems, Warwick, July 1981*, pages 54–57, 1981.

[Nat96] National Academy of Sciences. *Statistical Software Engineering*. National Academy Press, Washington, D.C., 1996.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice–Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[SB96a] J. Symanzik and A.L. Baker. Formalized Data Flow Diagrams and Their Relation to Other Computational Models. Technical Report 96–20, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, December 1996. Available by anonymous ftp from ftp.cs.iastate.edu or by e–mail from almanac@cs.iastate.edu.

[SB96b] J. Symanzik and A.L. Baker. Non–Atomic Components of Data Flow Diagrams: Stores, Persistent Flows, and Tests for Empty Flows. Technical Report 96–21, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, December 1996. Available by anonymous ftp from ftp.cs.iastate.edu or by e–mail from almanac@cs.iastate.edu.

[SB96c] J. Symanzik and A.L. Baker. Subclasses of Formalized Data Flow Diagrams: Monogeneous, Linear, and Topologically Free Choice RDFD's. Technical Report 96–22, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, December 1996. Available by anonymous ftp from ftp.cs.iastate.edu or by e–mail from almanac@cs.iastate.edu.

[TP89] T.H. Tse and L. Pong. Towards a Formal Foundation for DeMarco Data Flow Diagrams. *The Computer Journal*, 32(1):1–12, February 1989.

[War86] P.T. Ward. The Transformation Schema: An Extension of the Data Flow Diagram to Represent Control and Timing. *IEEE Transactions on Software Engineering*, SE–12(2):198–210, 1986.

[WBL93] T. Wahls, A.L. Baker, and G.T. Leavens. An Executable Semantics for a Formalized Data Flow Diagram Specification Language. Technical Report 93–27, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, November 1993. Available by anonymous ftp from ftp.cs.iastate.edu or by e–mail from almanac@cs.iastate.edu.

[WBL94] T. Wahls, A.L. Baker, and G.T. Leavens. The Direct Execution of SPECS–C++: A Model–Based Specification Language for C++ Classes. Technical Report 94–02b, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, Iowa 50011, February

1994, revised, November 1994. Available by anonymous ftp from ftp.cs.iastate.edu or by e–mail from almanac@cs.iastate.edu.

[WM85a] P.T. Ward and S.J. Mellor. *Structured Development for Real–Time Systems*, Volume 1: Introduction and Tools. Yourdon, Inc., New York, New York, 1985.

[WM85b] P.T. Ward and S.J. Mellor. *Structured Development for Real–Time Systems*, Volume 2: Essential Modeling Techniques. Yourdon, Inc., New York, New York, 1985.

[You89] E. Yourdon. *Modern Structured Analysis*. Yourdon Press Computing Series. Prentice–Hall, Englewood Cliffs, New Jersey, 1989.