Conformance of Distributed Systems

Maximilian Frey [1], Bernd-Holger Schlingloff [2, 3]

[1] O_2 (Germany) GmbH & Co. OHG, Georg-Brauchle-Ring 23-25,
80992 Munich, Germany
Maximilian.Frey@o2.com
[2] Humboldt-Universität zu Berlin, Institut für Informatik,
Rudower Chaussee 25, 12489 Berlin
[3] Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik FIRST,
Kekuléstr. 7, 12489 Berlin
Holger.Schlingloff@FIRST.FhG.DE

Abstract. This paper introduces a new conformance relation between a specification and an implementation of a distributed system. It is based on a local view which allows to avoid or reduce the state explosion problem. The conformance relation is defined via Petri nets and shows not only equivalence between transitions but also equivalence between local states. This equivalence depends on the structural properties of the Petri net and is independent of any specific initial marking. We compare our notion of conformance to classical ones and give model checking and test case generation algorithms for it.

1 Introduction

When testing telecommunication systems the testing devices, implementations and specifications usually are distributed systems. Testing of telecommunication systems means testing at different layers. Layer 1 or layer 2 according to the ISO reference model can be tested with purely sequential models, since there is only one end node. When testing higher layers (for example layer 3 or the network layer), at least the testing architecture is distributed.

Here is a short example in mobile communication to demonstrate this point. One specific interface within mobile GSM networks is the A-interface [GSM03.02]. This interface is positioned between the access network (also called BSS) and the core network. In order to test the correct implementation of an MSC (core network switch), the A-interface is of utmost importance, since all calls of the mobile customers are handled by that interface. A typical scenario is depicted in Figure 1 on the following page. Layer 2 of the A-interface is implemented by the SCCP (see reference [SCCP]) which is an SS7 protocol. The MSC is an end node for the SSCP; therefore, distributed testing is not necessary. It is sufficient to use a single point of control and observation (PCO). Layer 3 of the A-interface contains the call control protocol [GSM04.08].

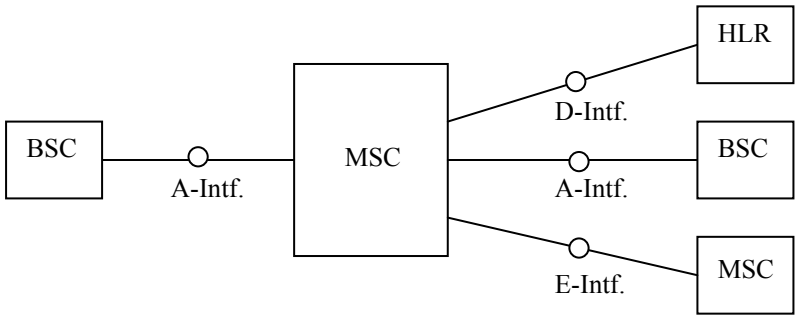

Fig. 1. Interfaces of the MSC

For testing the Call Control (CC) at least two different parties, the calling and the called party, are necessary. If the interfaces for the parties are different we have a protocol interworking scenario. For example interworking occurs between mobility management (MM) and MAP [MAP] at the D-interface or between CC and the ISUP [ISUP] on the E-interface. In that case the implementation is usually assumed to be a black box.
Extending the interworking from one node of the network to subnetworks like the access network or the core network, interworking between different nodes and a distributed implementation occurs. If new nodes or new versions of software are introduced in the network, integration tests in a distributed implementation must be performed. In that case the implementation can not be modelled as a black box, since the interfaces between the different nodes are traceable.

Distributed testing environments for telecommunication systems have been introduced some time ago. In TTCN-2 [TTCN-2] distributed testing components can be statically generated. Additionally, in TTCN-3 [TTCN-3] dynamic generation of distributed testing components is introduced. In [1999_GSBWL] a concept for a distributed testing environment based on TTCN and CORBA has been proposed.

Often, the specification of a telecommunication system refers to distributed parts. For example, the specifications of the Mobile Application Part Protocol [MAP], the DSS1 Protocol [DSS1] or the ISUP Protocol [ISUP2] contain SDL-specifications. A specification in SDL [SDL] consists of processes communicating via unbounded queues. The protocol specifications are divided into one protocol machine for incoming and one for outgoing calls. Furthermore, there is a supervision process which specifies how the different protocol machines are interacting. Each protocol machine has an external queue and at least one queue shared with the supervision process. Most of the layer-3 protocol specifications such as DSS1 or GSM/UMTS CC contain messages which have only local meaning as well as messages which also have global meaning. For example, a *call proceeding* message has only local meaning to the protocol machine, while a *setup* message has global meaning and is transferred to the supervision process.

Thus, there are causal independent events like the sending of a *call processing* message as local answer to a *setup* message and the global sending of the *setup* message through the network.
Especially in testing of telecommunication applications it is necessary to deal with distributed systems. Most existing methods for generating tests for communicating systems use finite state machines (see for example [Yao Petrenko Bochmann 1993], [Anido Cavalli 95] [Lee Yannakakis 1996]). This modelling formalism has the disadvantage that it does not differentiate between nondeterminism and parallelism.

Definition 1: A finite state machine (FSM) M is a quintuple $M=(I,O,S,\delta,\lambda)$ where I,O, and S are finite and nonempty sets of *input symbols*, *output symbols*, and *states,* resp. $\delta: S \times I \rightarrow S$ is the *transition function* and $\lambda: S \times I \rightarrow O$ is the *output function*.

In validation methods based on FSMs, distribution and parallelism is handled by building the cartesian product, i.e., by generating global states. This leads to a state explosion problem. To define some kind of test coverage, FSM based methods often define a conformance relation between FSMs. Applying these methods to distributed systems, often a huge amount of similar test cases is generated, where many individual test cases represent just different interleavings of the same distributed testing scenario. The cause for these unnecessary interleavings is in the state explosion which occurs by generating global from sets of local states. The generation of global states leads to two sorts of high complexity: Firstly, by the generation of global states from sets of local states, and secondly, by the derivation of distributed test scenario from the various generated sequences (see [Petrenko Ulrich Chapenko 1998].

Some authors propose to use distributed test case generation. Mostly, the proposed methods define a distributed transition tour through the system (see, e.g., [Ulrich König 1999] [Jard 2001] [Kim Shin Janson Kang 1999], and others). Compared to the methods for FSMs, transition tour methods are not able to test the conformance of FSMs, because they do not test the equivalence of states (see [Ramalingam Das Thulasiraman 95].
To define methods which are effective for distributed systems and test more than transition tours, a specification of conformance for distributed models is necessary. A well known model for distribution and parallelism are Petri nets. To model communication, we introduce an extension of Petri nets to include input and output symbols. A a new notion of correctness, we define a conformance relation between these nets. This relation can be used for model checking and testing purposes. Since our conformance relation is entirely based on local states, we avoid both of the drawbacks mentioned above.

Our paper is organized as follows. In Section 2, we give some basic definitions. Then, in Section 3 we define executions, simulations and conformance on these extended models. In Section 4 we compare the conformance relation on extended Petri nets to the conformance relation on FSMs. In Section 5 we sketch how conformance can be verified by model checking, and how tests can be automatically generated from Petri net specifications. Finally we give some conclusions and hints for further work.

2 Modelling of Distributed Systems

In order to model a subset of a telecommunication network, we have to model different nodes. Thus, the system under test (SUT) is the network. For the observation of this SUT, ports are used. They allow to observe the message flow on the external interfaces between the nodes.
For testing purposes, we distinguish between two kinds of ports.

- Ports which are points of control and observation (PCOs). The testing environment is applying input symbols to the SUT and receiving output symbols from the SUT.
- Ports which are points of observation (POs). These ports are in between two components of the SUT for internal communication. The input symbols of one component are the output symbols of the other component. The testing environment can only observe these input/output symbols.

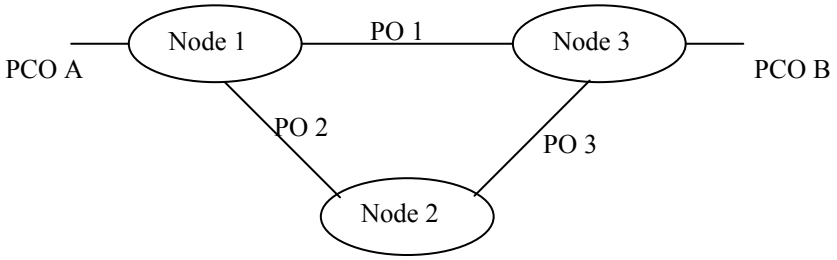

Fig. 2. Distributed system with PCOs and POs

Figure 2 shows an example for a distributed system with PCOs and POs. The system described contains three nodes. In a GSM system, PCO A and PCO B are PCOs on the CC protocol of an A-interface, Node1 and Node3 are MSCs and Node 2 is an HLR. PO 2 and PO 3 are POs on the MAP-Protocol of a C-interface.

Subsequently, we model distributes systems as the one above by Petri nets (cf. e.g. [Reisig 88]). The testing environment can send input symbols to the SUT only at PCOs. Therefore, Petri nets are extended with a set of input symbols and a set of output symbols. Similar to a transition in an FSM an event of a Petri Net receives an input symbol from the environment and generates an output symbol to the environment. In order to specify the input and output symbol for an event, two functions are added to the usual definition of a Petri Net.

Furthermode, PCOs and POs are modelled by sets of input and output symbols which occur at the specified port. Thus, the Petri net formalism additionally is extended by a set of PCOs and a set of POs. As a special case, the set of POs can be empty if the SUT consists of only one node.

Communication between different nodes is asynchronous, as within most telecommunication protocols. For any event *e* which has an output symbol of an PO there must exists a succecessor event *e'* such that the output symbol of *e* is the input symbol of *e'*. An internal communication via a PO has taken place. Between both events

is exactly one condition. It is not possible that a further event is between both, because that event would be an internal event which is not visible to any PO and PCO. In the same way each output and input event of one PO are connected.
Local (and also global) states of a extended Petri Net are *stable*, which means that they can only be left by an input from the environment. Similar to the FSM model, we require that all of our Petri nets are deterministic.

Definition 2: A extended Petri net P is a tuple P=(B,E,R,I,O,PCO,PO,ι,o), where

- B is a finite and nonempty set of *conditions*,
- E is a finite and nonempty set of *events*, (B∩E=∅)
- R ⊆ (B×E) ∪ (E×B) is the *flow relation*, where $x^\bullet=\{y \mid (x,y) \in R\}$ and $^\bullet x=\{y \mid (y,x) \in R\}$
- I is a finite set of *input symbols,*
- O is a nonempty, finite set of *output symbols*,
- PCO is a set of *ports* which are *points of control and observation*, i.e., a set of tuples (I_i, O_i) such that $I_i \subseteq I \setminus O$ and $O_i \subseteq O \setminus I$. PO is a set of ports which are *points of observation*, i.e., a set of sets (O_i) such that $O_i \subseteq I \cap O$.
- ι: E→ I is the *input function*,
- o: E→ O is the *output function*,
- For each $i \in I \setminus O$ there is exactly one PCO (I_i, O_i) such that $i \in I_i$. For each $o \in O \setminus I$ there is exactly one PCO (I_i, O_i) such that $o \in O_i$. For each $o \in I \cap O$ there is exactly one PO (O_i) such that $o \in O_i$.
- For any event e such that $o(e) \in I \cap O$ there exists a condition b and an event e' such that $^\bullet b = e$, $b^\bullet = e'$, and $\iota(e')= o(e)$.
  For any event e such that $\iota(e) \in I \cap O$ there exists a condition b and an event e' such that $b^\bullet = e$, $^\bullet b = e'$, and $\iota(e)= o(e')$.
- P is loop-free: $\forall x \in B \cup E\ (x^\bullet \cap {}^\bullet x = \emptyset)$
- P is deterministic: $\forall b \in B\ \forall e,e' \in b^\bullet\ (e \neq e' \rightarrow \iota(e) \neq \iota(e'))$

The distinction between PCOs and POs is for application purposes only; conceptually, the subsequent theory could also be based on one sort of ports.

Definition 3:
A *marking* of an extended Petri net P=(B,E,R,I,O,PCO,PO,ι,o) is any subset of B. Marking c' is *obtained* from c *by firing* event e (c[e⟩c') if $(^\bullet e \subseteq c \wedge c' = (c - {}^\bullet e) \cup e^\bullet)$. Marking c' is *reachable* from marking c (c[+⟩c'), if $\exists e_1, e_2, \ldots, e_n \in E\ \exists c_1, \ldots, c_{n-1} \subseteq B$ $(c[e_1\rangle c_1 \wedge c_1[e_2\rangle c_2 \wedge \ldots \wedge c_{n-1}[e_n\rangle c')$. We write c[*⟩c' if c=c' ∨ c[+⟩c'. A net is *one-safe from* c iff $\forall c' \subseteq B\ \forall e \in E\ (c[*\rangle c' \wedge {}^\bullet e \subseteq c' \rightarrow e^\bullet \cap c' = \emptyset)$.

Definition 4: An extended Petri net P=(B,E,R,I,O,PCO,PO,ι,o) is *strongly connected* if $\forall x,x' \in E \cup B: (x,x') \in R^*$.

Thus, an extended Petri net is strongly connected if all conditions and events are mutually reachable. We use nets to model reactive systems which are cyclic. Moreover, the communication at POs between nodes is in both directions. Therefore, all of our specifications are strongly connected.

The property of being strongly connected is not part of our definition of an extended Petri net. This is because we model the execution of Petri Nets by an unfolding of this net, which again is a net. Obviously an unfolding is not cyclic.

Also, the notion of *initial marking* is not part of our definition of a net. This is because we are aiming at a structural morphism between nets independent of the actual starting state. Each subnet is supposed to model a separate process, executing continuously and synchronizing by message passing.

Extended Petri nets can be used to model much more general systems than the telecommunication networks described above. To be closer to these, Petri nets would have to be built from components. For each PCO and each side of an PO a sequential, strongly connected component net would have to exist. However, such a restriction is not necessary for our subsequent considerations.

3 Conformance of Extended Petri Nets

To define a conformance relation on extended Petri Nets, we reconsider the conformance relation on FSMs. The execution of an FSM is a sequence of transitions observable by an input and output symbol each. Since our FSMs are deterministic, for any initial state and sequence of input signals exactly one execution of the FSM can be observed by the according sequence of output signals. The usual conformance relation between FSMs is a relation between states which means that not only the same transitions and transition paths must exist in two conforming FSMs but also the corresponding states are equivalent. To check whether a state is the inital state of an FSM is a state verification problem. If the implementation is considered as a black box, this can not be determined by testing [Lee Yannakakis 1996].

Definition 5: Let $M'=(I,O,S',\delta',\lambda')$ and $M=(I,O,S,\delta,\lambda)$ be two FSMs.

- For an initial state s_1 an input sequence $i_1, i_2, \ldots, i_n$ takes the FSM to states $s_{i+1}=\delta(s_i, i_i)$ and generates a output sequence $\lambda(s_1, i_1), \lambda(s_2, i_2), \ldots, \lambda(s_n, i_n)$.
- A state $s \in S$ is equivalent to a state $s' \in S$ if all input sequences M starting at s and M' starting at s' generate the same output sequences.
- Assume that M models an implementation and M' a specification. M conforms to M', if
  (1) For each state s in M there is a state s' in M' such that s is equivalent to s',
  (2) For each state s' in M' there is a state s in M such that s is equivalent to s'

Often, only part (1) or part (2) of the third condition is used, yielding a refinement relation. For FSMs, requiring (1) and (2) amounts to isomorphism of the corresponding minimal machines. This does not hold for more general computational models. We do not model executions of extended Petri Nets by sequences, but by causal nets

which preserve the partial ordering of causality in the original net. Here, conformance is more differentiating.

Definition 6: An extended causal net is an extended Petri net $K=(B_K,E_K,R_K,I_K,O_K,PCO_K, PO_K,t_K,o_K)$ where $\forall b \in B_K((|b^\bullet| \leq 1) \wedge (b,b) \notin R_K^+)$.

When generating a transition system from a net, each reachable marking represents a state. For a net with *n* conditions, there might be up to 2^n reachable markings. This is the so-called state explosion problem. We want to avoid the exponential blow-up by generating executions which start with a single condition of the net. However, executions of a Petri net should not be deadlocking. Therefore, we let each execution start at a marking which contains the chosen condition, plus some extra conditions which are necessary to avoid deadlock. An example can be found in figure 3. Assume that we want to generate an execution starting with condition b_1. Then, we also have to add condition b_6 to the initial marking, because otherwise event e_6 and all successor events of it would never be enabled, and the net would deadlock at $\{b_3, b_4\}$. For each condition in the net, there might be several markings containing it from which the net is live. As a representative we choose the minimal marking with respect to execution of events. The respective marking for the condition b_6 in figure 3 is indicated by a dashed line.

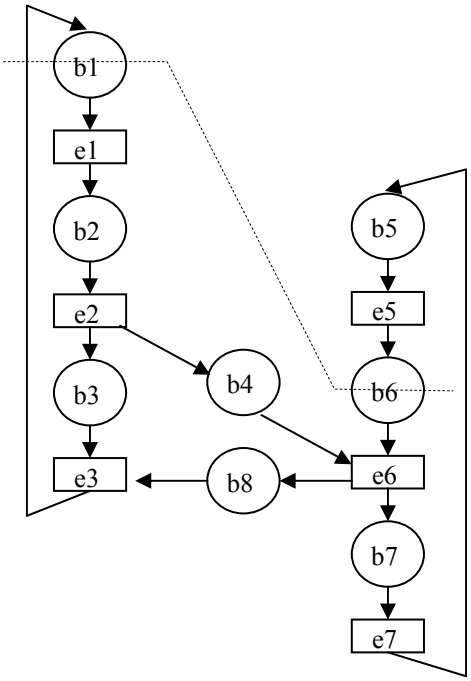

Fig. 3. Example of a strongly connected Petri net

Now, we define the execution of a net on a given input. In FSMs, an execution is determined by a sequence of input symbols supplied to the system. In extended Petri nets, there are several PCOs, where input symbols can be supplied independently and

in parallel. In order to model a distributed input for extended Petri nets, the following alternatives exist:

- A single sequence of input symbols is used. Symbols of different PCOs are totally ordered within this sequence.
- A partial order of input symbols is used, where input symbols of one PCO are totally ordered. Input symbols of different PCOs can be ordered or independent.
- There is no order on input symbols of different PCOs. The distributed sequence is represented by a set of sequences which contain a sequence of input symbols for each PCO.

We use the last of these alternatives, because it implies no additional causal relation on the execution of the Petri net. The other choices would lead to additional causal dependencies which are not covered in the specification net.

Definition 6: We say that SEQ=$\{seq_1,seq_2,...,seq_n\}$ is an *admissible input* for the net P=(B,E,R, I,O,PCO,PO,ι,o), where PCO contains *n* ports (I_i,O_i), if each seq_i is a non-empty and finite sequence of input symbols from I_i.

Let P be an extended Petri net P=(B,E,R, I,O,PCO,PO,ι,o). Each execution starts from a certain initial condition for one process. All nondeterministic choices for this process should be represented in the initial marking.

Definition 7: An initial marking of P for the condition b, denoted by c(b), is a set of conditions such that all events having b as precondition are enabled in c(b); i.e., $\forall e\in E\ (e\in b^\bullet \rightarrow {}^\bullet e\subseteq c(b))$.

Since we are aiming at a partial order model of program runs, we require that each execution of a net is a (contact-free) causal net.

Definition 8: Let P be an extended Petri net P=(B,E,R, I,O,PCO,PO,ι,o), let $b_I\in B$ be some condition from P, and let SEQ be an admissible input for P. Furthermore, let K be an extended causal net $K=(B_K,E_K,R_K, I,O,PCO,PO,\iota_K,o_K)$ with the same input and output alphabets and PCOs and POs as P. We say that K is an *execution of P starting at b_I according to SEQ* (denoted by $K\in[P,b_I,SEQ\rangle$) if a homomorphism h: $K\rightarrow P$ and an initial marking $c(b_I)$ exist such that

- $h(B_K)=B$
- $h(E_K)=E$
- $\forall e\in E_K\ ((h(e^\bullet)=h(e)^\bullet)\wedge({}^\bullet h(e)={}^\bullet(h(e)))$
- $\forall b,b'\in B_K\ (((b',b)\notin R_K^* \wedge (b,b')\notin R_K^*)\rightarrow h(b)\neq h(b'))$
- $\forall b\in B_K\text{-}\ h(c(b_I))\ \ (|{}^\bullet b|=1)$
- $\forall b\in h(c(b_I))\ (|{}^\bullet b|=0)$.
- $\forall e\in E_K\ (h(\iota_K(e))=\iota(h(e)))$
- $\forall e\in E_K\ (h(o_K(e))=o(h(e)))$
- $(\exists e_1, e_2, ...\in E\ \exists c_1, c_2, ...\subseteq B\ \exists i_1, i_2, ...\in I\ ({}^\bullet e_1\subseteq c(b)\wedge c_1=(c(b)\text{-}\ {}^\bullet e_1)\cup e_1^\bullet\wedge\iota(e_1)=i_1\wedge {}^\bullet e_i\subseteq c_{i-1}\wedge c_i=(c_{i-1}\text{-}\ {}^\bullet e_i)\cup e_i^\bullet\wedge\iota(e_i)=i_i\wedge\forall seq\in SEQ\ (\exists(I',O')\in PCO\ (\forall x\in seq\ (x\in I')\wedge i_1, i_2, ...|_{I'}=seq))$

Thus, the execution of an extended Petri net is a causal net which describes the causal dependencies between conditions and events together with the input symbols which have been consumed by the events and the output symbols which have been generated by the events. An algorithm to construct executions from nets is given in Section 5 below.

In the definition of conformance for FSMs the equivalence of states is based on the equality of generated output sequences. For extended Petri nets, we model executions by causal nets; thus, conformance will be based on a relation between causal nets. In the following Definition 9, we propose two different alternatives. In both cases a homomorphism between the execution of the specification and the execution of the implementation is used. Also the input and output function is preserved by the homomorphism.

Definition 9: Let $Ki=(B_{Ki},E_{Ki},R_{Ki},I,O,PCO,PO,\iota_{Ki},o_{Ki})$ and $Ks=(B_{Ks},E_{Ks},R_{Ks},c_{Ks}$ $I,O,PCO,PO,\iota_{Ks},o_{Ks})$ be two extended causal nets with the same input and output symbols and the same ports. Intuitively, Ki is an execution of some implementation I, and Ks is an execution of some specification S.

- Ki is *weakly simulating* Ks if a mapping h: Ks → Ki exists such that
  $\forall\, x,x' \in E_{Ks} \cup B_{Ks}\ ((x, x') \in R_{Ks} \rightarrow (h(x), h(x')) \in R_{Ki}) \wedge$
  $\forall\, e \in E_{Ks}\ (o_{Ks}(e) = o_{Ki}(h(e)) \wedge \iota_{Ks}(e) = \iota_{Ki}(h(e)))$
- Ki is *strongly simulating* Ks if a mapping h: E_{Ks} → E_{Ki} exists such that
  $\forall\, e,e' \in E_{Ks} \cup B_{Ks}\ ((x, x') \in R_{Ks} \leftrightarrow (h(x), h(x')) \in R_{Ki}) \wedge$
  $\forall\, e \in E_{Ks}\ (o_{Ks}(e) = o_{Ki}(h(e)) \wedge \iota_{Ks}(e) = \iota_{Ki}(h(e)))$

The difference between the weak and strong simulation relation is that the strong simulation relation preserves the causal relation of the specification. The weak simulation relation allows that the implementation contains additional causal dependencies, but no additional events. An example is given in figure 4, where the execution (b) is weakly, but not strongly simulating (a).

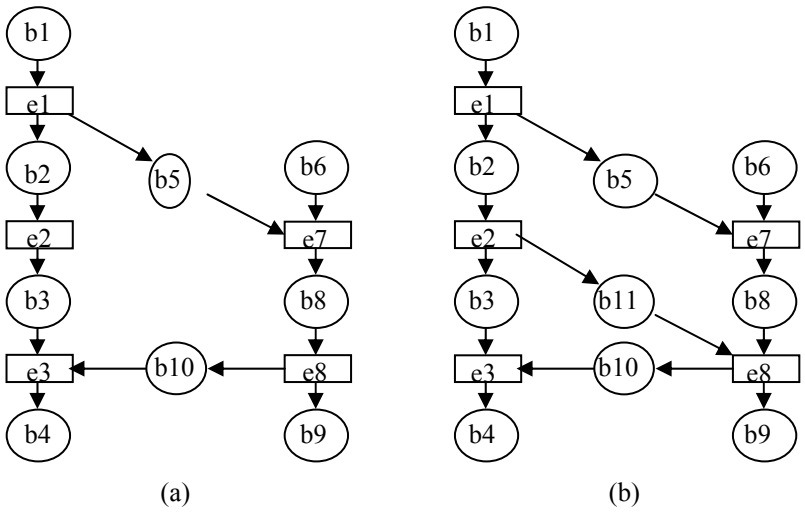

Fig. 4. Run of the specification (a) and the implementation (b) for the same input sequences

Conformance between implementation and specification is defined similar as for FSMs. The conformance relation is based on (weak) simulation between conditions.

Definition 10:
Let $I=(B_I,E_I,R_I,I,O,PCO,PO,\iota_I,o_I)$ and $S=(B_S,E_S,R_S,I,O,PCO,PO,\iota_S,o_S)$ be strongly connected extended Petri nets (the implementation and specification, respectively).

- We say that condition $b_I \in B_I$ is *(weakly) simulating* condition $b_S \in B_S$ if for all admissible inputs SEQ and execution K_I and K_S such that $K_I \in [I,b_I,SEQ\rangle$, $K_S \in [S,b_S,SEQ\rangle$ it holds that $K_I \in$ is (weakly) simulating K_S.
- I *(weakly) conforms to* S if
    1. $\forall\, b_S \in B_S\, (\exists\, b_I \in B_I\, (b_I$ is (weakly) simulating $b_S))$, and
    2. $\forall\, b_I \in B_I\, (\exists\, b_S \in B_S\, (b_I$ is (weakly) simulating $b_S))$

For functional correctness, it is necessary that an implementation has the same causal dependencies as the specification. Otherwise the implementation could contain a race condition, which is not allowed by the specification. This is an error which occurs frequently in practical applications and is very difficult to detect. On the other side, if an implementation execution has more dependencies than the corresponding specification execution, this indicates that the degree of parallelism is reduced in the implementation. This might cause a performance problem, but does not influence the functional correctness. Whereas the specification and verification of performance is beyond the scope of the present paper, additional dependencies in the implementation leading to deadlock or similar errors can be found with our weak simulation relation. In this case the I/O-behavior of the implementation would be different to the behavior of the specification for at least one state and one set of input sequence. Thus, (weak) simulation is sufficient for testing of correct functionality and is used in the above definition.

4 Conformance in Distributed and Sequential Models

A conformance relation based on a distributed and parallel model has been introduced above. In this section this relation is compared with the conformance relation on FSMs in two ways. We ask the following questions:

1. Are both confomance relations equivalent for sequential systems?
2. Is the conformance relation on extended Petri nets equivalent to the conformance relation on the global FSM of the Petri net?

For the first question we define a subclass of extended Petri nets, sequential Petri nets which can describe sequential systems only. This class will be equivalent to FSMs. Therefore the number of PCOs is one and no parallelism is generated.

Definition 11: An extended Petri Net $P=(B,E,R,I,O,PCO,PO,\iota,o)$ is *sequential*, if $\forall\, e \in E\, (|{}^\bullet e| \leq 1$ and $|e^\bullet| \leq 1)$ and $\forall\, b \in B\, (|c(b)| \leq 1)$ and $|PCO|=1$ and $|PO|=0$.

Theorem 1 shows that the class of sequential Petri nets is equivalent to FSMs.

Theorem 1: Strongly connected sequential Petri Nets and strongly connected FSM are equivalent.
Proof: Since |PCO|=1 and |PO|=0, we have PCO={(I,O)}. Since $\forall$ e$\in$E ($|^\bullet e|\leq 1$ and $|e^\bullet|\leq 1$) and $\forall$ b$\in$B ($|c(b)| \leq 1$), all events can be used to define relations λ and δ: $\forall$ e$\in$E $\forall$ b,b'$\in$B (((b,e) $\in$R and (e,b') $\in$R) $\rightarrow$ (δ(b,ι(e))=b' $\wedge$ λ(b,ι(e))=o(e))). This way we have defined an FSM with (I,O,B,λ,δ).
Vice versa, from an FSM a Petri net can be generated in a similar way.

In order to show that the conformance relation on FSMs is equivalent to the conformance relation on sequential Petri nets, we first have to show that executions of sequential Petri nets can be represented by sequences of output symbols. The second step of the proof is to show that the weak simulation relation is equivalent to the equality on output sequences.

Lemma 1: Each execution of strongly connected sequential Petri Nets P starting at a condition b and according to a set SEQ of sequences of input symbols [P,b,h,SEQ$\rangle$ represents a sequence of output symbols.

Proof:
K=(B_K,E_K,R_K,I,O,PCO,PO,ι_K,o_K)= [P,b,h,SEQ$\rangle$ is an execution of a sequential Petri Net P=(B,E,R,I,O,PCO,PO,ι,o). Since |PCO|=1, SEQ contains exacly one sequence. Since P is sequential and $\forall$ e$\in$E ($|^\bullet e|\leq 1$ and $|e^\bullet|\leq 1$), it is valid that $\forall$ e$\in$ E_K ($|^\bullet e|\leq 1$ and $|e^\bullet|\leq 1$).Because P is strongly connected, it is valid that $\forall$ e$\in$ E_K ($|^\bullet e|=1$ and $|e^\bullet|=1$) Because K is [P,b,h,SEQ$\rangle$, it is valid, that $\forall$ s$\in$((B_K- h(c(b)))$\cup E_K$) ($|^\bullet s|=1$ and $|^\bullet s|=1$ and (s,s) $\notin R_K^+$). $\forall$ s$\in$ h(c(b)) ($|^\bullet s|=0$ and $|^\bullet s|=1$ and (s,s) $\notin R_K^+$). K describes a sequence h(c(b)), e_1, b_1, e_2, b_2, e_3, b_3, e_4, ... The output sequence is o(e_1), o(e_2), o(e_3), o(e_4), ...

Lemma 2: For two strongly connected sequential Petri Nets I=(B_I,E_I,R_I, I,O,PCO,PO,ι_I,o_I) and S=(B_S,E_S,R_S, I,O,PCO,PO,ι_S,o_S) the condition $b_I \in B_I$ is weakly simulating the condition $b_S \in B_S$ if and only if for all SEQ, [I,b_I,h_I,SEQ$\rangle$ and [S,b_S,h_S,SEQ$\rangle$ are representing the same output sequence.

Proof: "$\rightarrow$"
From Lemma1: [I,b_I,h_I,SEQ$\rangle$ is represented by $h_I(c(b_I))$, e_{I1}, b_{I1}, e_{I2}, b_{I2}, e_{I3}, ..., and [S,b_S,h_S,SEQ$\rangle$ is represented by $h_S(c(b_S))$, e_{S1}, b_{S1}, e_{S2}, b_{S2}, e_{S3}, ... Since b_I is weakly simulating b_S for all SEQ [I,b_I,h_I,SEQ$\rangle$ is weakly simulating [S,b_S,h_S,SEQ$\rangle$. Because S and I are sequential, all possible SEQ contain only one sequence of input symbols. [I,b_I,h_I,SEQ$\rangle$=(B_{Ki},E_{Ki},R_{Ki},I,O,PCO,PO,ι_{Ki},o_{Ki}) and [S,b_S,h_S,SEQ$\rangle$ =(B_{Ks}, E_{Ks}, R_{Ks}, c_{Ks}, I, O, PCO, PO, ι_{Ks}, o_{Ks}) and it exists an homomorphism h':[S,b_S,h_S,SEQ$\rangle$ $\rightarrow$ [I,b_I,h_I,SEQ$\rangle$ such that h'(E_{Ks})= E_{Ki} and $\forall$ x,x'$\in$ $E_{Ks}\cup B_{Ks}$ ((x, x') $\in$ R_{Ks} $\rightarrow$ (h(x), h(x')) $\in$ R_{Ki}) and $\forall$ e$\in E_{Ks}$ (o_{Ks}(e)= o_{Ki}(h(e)) $\wedge$ ι_{Ks}(e)= ι_{Ki}(h(e))). h([S,b_S,SEQ$\rangle$) is represented by h'($h_S(c(b_S))$), h'(e_{S1}), h'(b_{S1}), h'(e_{S2}), h'(b_{S2}), h'(e_{S3}), h'(b_{S3}), h'(e_{S4}), ... and the output sequence of [I,b_I,h_I,SEQ$\rangle$ is the same as that of [S,b_S,h_S,SEQ$\rangle$

"←"
From Lemma1: $[I,b_I,h_I,SEQ\rangle$ is represented by $h_I(c(b_I))$, e_{I1}, b_{I1}, e_{I2}, b_{I2}, e_{I3},..., and $[S,b_S,h_S,SEQ\rangle$ is represented by $h_S(c(b_S))$, e_{S1}, b_{S1}, e_{S2}, b_{S2}, e_{S3},... . $[I,b_I,h_I,SEQ\rangle = (B_{Ki},E_{Ki},R_{Ki},I,O,PCO,PO,\iota_{Ki},o_{Ki})$ and $[S,b_S,h_S,SEQ\rangle = (B_{Ks},E_{Ks},R_{Ks},c_{Ks}, I,O,PCO,PO,\iota_{Ks},o_{Ks})$. The output sequences of $[I,b_I,h_I,SEQ\rangle$ and $[S,b_S,h_S,SEQ\rangle$ are the same: $o(e_{Ii}) = o(e_{Si})$. Because I and S are sequential Petri nets, |PCO|=1 and SEQ contains exactly one element seq. That implies that $\iota(e_{Ii}) = \iota(e_{Si})$. A homomorphism h': $[S,b_S,h_S,SEQ\rangle \rightarrow [I,b_I, h_I,SEQ\rangle$ can be defined as follows: $h'(e_{Si})= e_{Ii}$, $h'(b_{Si})= b_{Ii}$, $h'(h_S(c(b_S)))= h_I(c(b_I))$. This homomorphism defines that $[I,b_I,h_I,SEQ\rangle$ is weak simulating $[S,b_S,h_S,SEQ\rangle$ and b_I is weak simulating b_S.

Theorem 2: The conformance relation on strongly connected FSMs is the same as that on strongly connected sequential Petri nets.
Proof: Follows directly from Lemma 2 and Theorem 1.

Thus, the answer to the first question from above is affirmative. For answering the second question we have to define the global FSM of an extended Petri net. For communicating finite state machines (CFSM) the Cartesian product of the states is used to generate a FSM. On this FSM the conformance relation is defined.

Definition 12: The global FSM $G(P)=(I,O,S,\lambda,\delta)$ of an extended strongly connected Petri Net $P = (B,E,R,I,O,PCO,PO,\iota,o)$ contains the set S of all sets $c \subseteq B$, such that (B,E,R,c) is contact free, there exists an event e with ${}^\bullet e \subseteq c$, and $\delta(c,i)=c'$ and $\lambda(c,i)=o'$ if it exists an event e with ${}^\bullet e \subseteq c$, $e^\bullet$ and c' are disjoint, $c'=(c-{}^\bullet e) \cup e^\bullet$ and $\iota(e)=i$ and $o(e)=o'$.

We demonstrate the difference between the conformance relation on extended Petri nets and global FSM of extended Petri nets by an example. Figure 5 shows a specification and an implementation. Both extended Petri-Nets have two PCOs defined by ({i1},{o1}) and ({i2},{o2}). The input and output function is defined by $\iota(e)=i1$ $o(e)=o1$ for e= e1,e2,e3 and $\iota(e)=i2$ $o(e)=o2$ for e= e7,e8.
The initial markings of the specification are for condition b1 {b1, b6}, for condition b2 {b2, b8}, for condition b3 {b3, b10, b6}, for condition b5 {b5, b6, b3}, for condition b6 {b6, b5, b3}, for condition b8 {b8, b3} and for condition b10 {b10, b3, b6}.

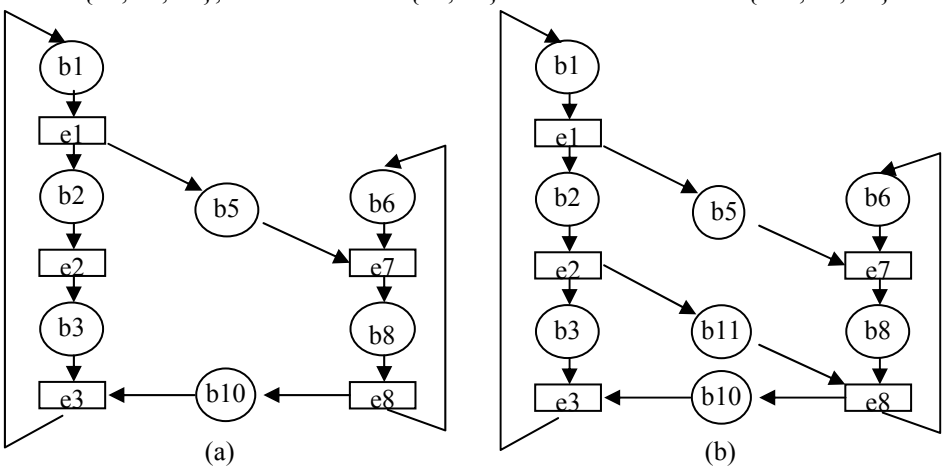

Fig. 5. Extended Petri net of the specification (a) and of the implementation (b)

The initial markings of the implementation are for condition b1 {b1, b6}, for condition b2 {b2, b8}, for condition b3 {b3, b10, b6}, for condition b5 {b5, b6,b3, b11}, for condition b6 {b6, b5, b3, b11}, for condition b8 {b8, b3, b11}, for condition b10 {b10, b3, b6} and for condition b11 {b11, b8, b3}.

Thus for each condition b in the specification a condition b' in the implementation exists with b=b' and b' is weakly simulating b. Also for each condition b'=b1, b2, b3, b5, b6, b8, b10 a condition b in the specification exists with b=b' and b' is weakly simulating b. Condition b11 is weakly simulation condition b8. The implementation Petri net conforms to the specification Petri net.

The corresponding global FSMs of the extended Petri nets of Figure 5 are shown in figure 6.

In state b2b8 in the specification the sequence of input symbols in_seq=i2, i1, i1, i1, i2, i1 produces the output sequence out_seq=o2, o1, o1, o1, o2, o1. There is no state in the implementation which will produce out_seq when in_seq is applied. The FSMs do not conform even though the Petri nets conform. That means conformance of global FSM of Petri nets is not equivalent to conformance on Petri nets. One reason for this is the asymmetry of the weak simulation relation on the executions of the extended Petri nets. A discussion which conformance relation is more "intuitive" as well as a comparison of the expressiveness of the two notions will be given in a subsequent paper.

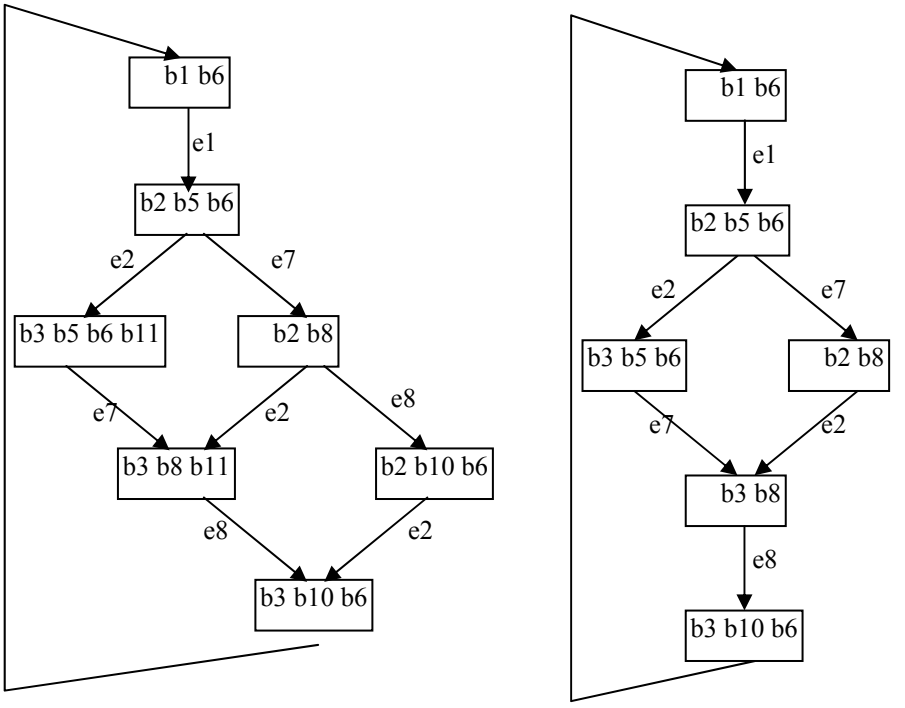

Fig. 6. Global FSMs of the specification (a) and of the implementation (b) as defined in figure 5.

5 Model Checking and Distributed Test Generation

So far we have assumed that models both for the specification and for the implementation are given. In this case, conformance checking is a special form of model checking. In particular, since both specification and implementation are deterministic, we can use a classical partition refinement algorithm to establish conformance.

Let H_0 be the relation consisting of all pairs $(b_I,b_S) \in B_I \times B_S$. H_{i+1} is constructed from H_i as follows:

$(b_I,b_S) \in H_{i+1}$ iff

1. $(b_I,b_S) \in H_i$, and
2. $\forall\, e_I \in {}^\bullet b_I,\ e_S \in {}^\bullet b_S\ (\iota_{Ki}(e_I) = \iota_{Ks}(e_S) \rightarrow o_{Ki}(e_I) = o_{KS}(e_S))$, and
3. $\forall\, e_S \in {}^\bullet b_S\ \forall\, b_S' \in e_S^\bullet\ \exists\, e_I \in {}^\bullet b_I\ \exists\, b_I' \in e_I^\bullet : (b_I',b_S') \in H_i$

Since we are dealing with finite Petri nets, the iteration must reach a fixed point. Let H be the relation reached upon stabilization. Then I conforms to S if

1. $\forall\, b_S \in B_S\ \exists\, b_I \in B_I : (b_I,b_S) \in H$, and
2. $\forall\, b_I \in B_I\ \exists\, b_S \in B_S : (b_I,b_S) \in H$.

Often, the internals of the implementation are not accessible in a systems validation process. In this case, we have to resort to black box testing techniques. In our approach, each execution of the specification can serve as the basis for the generation of test cases. We use the following algorithm to generate executions from a given extended Petri net P:

- Start with an arbitrary condition b and let c(b) be $\cup\{{}^\bullet e \mid e \in b^\bullet\}$
- The initial part of the execution is a copy of all conditions in c(b)
- Put a mark on all conditions in c(b)
- Repeat indefinitely
  - Choose a maximal set of events which are either enabled in P, or can be enabled by putting a token on a condition which is not marked, such that the inputs of these events contain at most one input from each PCO and PO, respectively.
  - Put a mark on all conditions which have received a token, as well as on all conditions in the pre- and postset of an enabled transition.
  - Fire the chosen events in P, and extend the execution by appending a copy of all chosen events and their postsets to it.

During the execution of the Petri Net the initial marking c(b) is generated on the fly. Starting with the marking containing all conditions in the preset of the succeeding events of b, conditions are added when a state is reached in which no event can be fired. Only conditions of the preset of an event are added such that the event get fireable. Each condition can only be added once to the initial state. In comparision to

FSMs, an execution is described by a combination of the input sequence and the output sequence together with the reached states of the FSM.

Since each execution represents a class of different interleavings, it is sufficient to generate only a few representatives from it which cover the intended behavior. The particular interleaving is chosen according to some specific heuristics, e.g., firing all enabled events always according to some specific ordering. Another alternative is to map the execution to a TTCN test case description and let the TTCN scheduler do the linearization of concurrent events.

In general, this technique can reduce the number of test cases by an exponential factor. If each potential error is preserved by this reduction technique, then the testing coverage in the specification is the same as that without partial order reduction; i.e., the same test termination and measurement criteria can be used.

6 Conclusion and Further Work

We have proposed a conformance relation between specification and implementation of distributed systems. This relation is based on a local view of the distributed system which allows to avoid or reduce the state explosion problem. As distributed model we use an extension of one-safe Petri nets. The conformance relation on extended Petri nets is based not only on the equivalence between transitions but also on the equivalence between conditions which are local states. The equivalence between conditions depends only on the structural properties of the Petri net and is independent of any specific initial marking.

We compared the conformance relation on Petri nets with the conformance relation on FSMs. The result is that for sequential systems both conformance relations are equivalent. For distributed systems the conformance relations are not equivalent.

Further work includes a more detailed comparison between both conformance relations for distributed systems. In particular, we plan to investigate a variant of the conformance relation on Petri nets using the strong simulation relation and compare it with the relation on FSMs. Furthermore, we plan to adopt methods for generation of test to the conformance relation on Petri nets and to define a coverage measure which determines the extent to which a black box implementation conforms to a specification. Finally, we will implement concrete testcases in TTCN-3 for telecommunication networks. For this, a concrete algorithm for the generation of a module from causal nets has to be developed. For practical applications in telecommunications, we also plan to develop methods to handle the different parameters and information elements sent within the messages across the network and to incorporate these into the generation of TTCN-3 testcases.

References

[GSM03.02] ETSI: ETSI TS 100 522 Digital cellular telecommunications system (Phase 2+): Network architecture (GSM 03.02 version 7.1.0 Release 1998): 2000

[SCCP] ITU-T: Recommendations Q.711-Q.714 Specifications of Signalling System No. 7 - Signalling connection control part: 1996

[GSM04.08] ETSI: ETSI TS 100 940 Digital cellular telecommunications system (Phase 2+); Mobile radio interface layer 3 specification (3GPP TS 04.08 version 7.17.0 Release 1998): 2002

[MAP] ETSI: ETSI TS 129 002 Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Mobile Application Part (MAP) specification (2GPP TS 29.002 version 4.50 Release 4): 2001

[ISUP] ITU-T: Recommendations Q.761-Q.764 Signalling System No. 7 – ISDN user part signalling procedures: 1999

[TTCN-2] ISO. ISO/IEC 9646-3, Tree and Tabular Combined Notation (TTCN), Second Edition (1997)

[TTCN-3] ETSI: ETSI ES 201 873 Methods for Testing and Specification (MTS); The Tree and Tabular Combined Notation version 3: 2001

[1999_GSBWL] Vassiliou-Gioles, T., Schieferdecker, I., Born, M., Winkler, M., Li, M.: Configuration and Execution Support for Distributed Tests. In: Csopaki, G., Dibuz, S., Tarnay, K. (eds): Testing of Communicating Systems. Kluwer Academic publishers. Boston, Dordrecht, London (1999) 61-76

[DSS1] ITU-T: Recommendation Q.931 Digital Subscriber Signallingsystem No. 1 (DSS 1) – ISDN User-Network Interface Layer 3 Specification For Basic Call Control: 1998

[ISUP 2] ITU-T: Recommendation Q.764-Annex H Specification of Signalling System No.7 – ISDN User Part Signalling Procedures Annex H: State Transition Diagrams: 1995

[SDL] ITU-T: Recommendation Z.100 Languages for Telecommunications Applications – Specification and Description Language, 1999

[Yao Petrenko Bochmann 1993]Yao, M., Petrenko, A., v. Bochmann, G.: Conformance Testing of Protocol Machines without Reset. In: Danthine, A., Leduc, G., Wolper, P. (eds): Protocol Specification, Testing and Verification, XIII.Elsevier Science Publishers B. V. (North-Holland) (1993) 241-253

[Anido Cavalli 95] Anido, R., Cavalli, A.R.: Guaranteeing full fault coverage for UIO-based testing methods. In: Proceedings of the 8th Int. Workshop on Protocol Test Systems, Evry, France (1995) 221-236

[Lee Yannakakis 1996] Lee, D., Yannakakis, M.: Princliples and Methods of testing Finite State Machines – A Survey. Proceedings of the IEEE. Vol. 4 (8), 1996, 1090-1123

[Petrenko Ulrich Chapenko 1998] Petrenko, A., Ulrich, A., Chapenko, V. :Using partial-orders for detecting faults in concurrent systems. In: Proceedings of Workshop on Testing of Communicating Systems (IWTCS'98), Russia, 1998

[Ulrich König 1999] Ulrich, A., König, H.: Architectures for Testing Distributed Systems. In: Csopaki, G., Dibuz, S., Tarnay, K. (eds): Testing of Communicating Systems. Kluwer Academic publishers. Boston, Dordrecht, London (1999) 93-108

[Jard 2001] Jard, C., Principles of Distributed Test Synthesis based on True-Concurrency Models. In: Schieferdecker, I., König, H., Wolisz, A.: Testing of Communicating Systems XIV. Kluwer Academic publishers. Boston, Dordrecht, London (2002) 301-316

[Kim Shin Janson Kang 1999] Kim, M., Shin, S.T., Chanson, S. T., Kang, S.: An Enhanced Model for Testing Asynchronous Communicating Systems. In: Formal Description Techniques and Protocol Specification, Testing and Verification, 19. IFIP (1999) 337-355

[Ramalingam Das Thulasiraman 95] Ramalingam, T., Das, A., Thulasiraman, K.: Fault detection and diagnosis capabilities of test sequence selection methods based on the FSM model. Computer communications, vol.18(2), 1995,113-122

[Reisig 88] Reisig, W.: Petri Nets. Springer 1988

[Tretmans 96] Test Generation with Inputs, Outputs and Repetitive Quiescence. Software--Concepts and Tools, 17(3):103-120, 1996