Quantitative assessment of a peer-to-peer cooperative infrastructure using Stochastic Well-Formed Nets

Carlo Bellettini [a,1] Lorenzo Capra [a,2] Mattia Monga [a,3]

[a] *DICo - Dipartimento di Informatica e Comunicazione*
Università degli Studi di Milano, Italy

Abstract

Traditional support tools for software engineers, normally based on a client-server architecture, are unsuitable to deal with the new issues emerging from the current (and future) cooperative work scenarios (where connectivity is intrinsically transient, the number of interacting partners dynamically changes, etc.). This paper presents a quantitative assessment of a fully decentralized, peer-to-peer, cooperative infrastructure. Stochastic Well-Formed Nets (SWNs) modelling the new peer-to-peer architecture, and a traditional (client-server) one, are developed and analysed: we used SWNs for their ability to directly exploit the symmetries intrinsically present in the modelled systems, thus greatly reducing the complexity of the analysis. The main goal is to compare the impact of the two alternative protocols on the collaborative work. Together with the performance figures of interest, methodological issues concerning the choice of the most appropriate model abstraction level, the adoption of a compositional modelling approach, and the management of the model complexity are also discussed.

Key words: SWNs, peer-to-peer, configuration management

1 Introduction

Nowadays the Internet infrastructure is so pervasive that it is common that people connect their laptop computer from a range of different locations: office, home, the hotel hosting them for a conference, or the meeting room where they are working. This is sometimes called mobile computing and it forces designers of applications to take into account two new requirements: (1) users may connect to the network from arbitrary locations (usually with different network

[1] Email: carlo.bellettini@unimi.it
[2] Email: capra@dico.unimi.it
[3] Email: mattia.monga@unimi.it

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

addresses), and (2) they are not permanently connected. Thus, connectivity is intrinsically transient, and machine disconnection is not an exceptional case, but the normal way of operating.

Collaborative work over the Internet is often pursued by manipulating electronic artifacts that are shared with others by exploiting the mediation of some server. This scenario is still common and useful in several cases. However, since work activities are no more exclusive of office settings, users are often forced to work without any Internet connection. Thus, network applications that rely on servers are sometimes not desirable or even not feasible. In other words, the client/server approach is possible, and common, in all cases where a reliable and permanent network infrastructure is available to connect the participating nodes. However, in many cases, people would like to collaborate while they are supported through a much looser architecture, since they cannot or do not want to afford the cost of setting up a central server. The resulting reference architecture is a network of peers, each of which contributes to the overall logical structure in an equivalent way. Moreover, peers cannot be assumed to be always on-line, i.e., a peer is not always reachable by others. Since the network connection is intrinsically intermittent, peers may dynamically join and leave an ad-hoc community. They join it in impromptu meetings, where they synchronize their works.

We focussed our investigation on the collaborative work needed to produce software systems. Software developers typically collaborate by exchanging and sharing a number of files. Files are assigned to people according their responsibilities in the project. However, besides the person in charge of a file, several other collaborators sometimes need to view or modify it. In general, for each item we can identify the role of an *owner,* i.e., the individual who has created the artifact or who is in charge of carrying out the work on it. Moreover, there is a number of other collaborators involved in the project who need to manipulate items that are not under their direct control, i.e., artifacts they do not own. System designed to help software developers in keeping their work coherent and tracking the evolution of artifacts are called *configuration management tools.*

The underlying architecture of these tools affects the ongoing collaborative work. In fact, a server based solution forces people to keep their work consistent with the copy on the server machine. In traditional, client/server version management tools, two (or more) persons may work on the same artifact (a file): both are required to check-out the artifact from the server machine and after any modification it has to be accepted on the server machine by trying a commit (or check-in) operation. Different control policies for concurrent work are possible. Artifacts could be locked by the system while people are modifying them, in order to avoid concurrent modifications. Instead, *optimistic* concurrency control [9] means that if Alice and Bob are both working on `foo.c` and Alice commits her new version in the central repository, when Bob tries to check-in his own modified version he gets a conflict from the server and he

must merge his modifications with Alice's ones before trying a new check-in.

`PeerVerSy` [1] is a version management tool that allows users for freely accomplishing their computations and collaborative actions as check-in and check-out also when they are not able to communicate with the machine that holds the reference copy of an artifact. Concurrency control is basically optimistic and an automatic reconciliation step is performed when connection is established again, possibly arising conflicts. The client/server approach assumes the availability of the network infrastructure even in the frequent case that no concurrent work is done on a particular item. It is perfectly reasonable and desirable that one could check-in a file which is under his/her control if no other developers want to manipulate it. Similarly, check-out operations can be performed also when the latest version of an artifact is available somewhere– not necessarily on repository servers– but for example on the local file system or on the file system of any connected node.

In the `PeerVerSy` approach there are no well known servers. Instead, it is based on the notion of the *authority* for a set of items (it *owns* them), and the copy of an artifact owned by the authority is the *master* copy. In addition to the master copy, other peers can keep a local copy (*replica*) of the documents they do not own in order to allow users to work on them even when the authority is not reachable or when the peer is disconnected from the network. In fact, a user can perform both check-in and check-out operations also from the local copies of a document and the only difference between the master copy and a replica is that a check-in of a new version becomes definitive and available for all users only when the authority accepts the changes and updates the master copy.

Intuitively, the peer-to-peer protocol may affect the number of conflicts generated during the collaboration. The analysis described in this paper aims at comparing the performances of a server based optimistic protocol with the `PeerVerSy` protocol. In particular, we wanted to measure the impact of the underlying architecture on the frequency of merge operations. In our experience with the tool we found that `PeerVerSy` was extremely attractive for small (< 10 people) groups of developers. In fact, specially when several organizations are involved (as it is often the case in academics and research settings), is sometimes unsuitable to set up a server based configuration management tool: a solution that allows people for using their own machine, installing and configuring the program on their own responsibility is much more appealing for its improved flexibility. Moreover, users are happy to be able to freely cooperate wherever they want, and even work at home without any Internet connection. However, we wanted to be able to assess the impact of the approach on the work in a quantitative fashion, out by means of stochastic models. Although initially a simple simulation environment for `PeerVerSy` was set up [11] (on which a preliminary evaluation of the system was performed), we eventually preferred modeling essentially because, notwithstanding approximations intrinsic in both approaches, due to unavoidable abstractions in the system

representation, whenever the analytical model solution is feasible it is more accurate and allows a cheaper and more flexible assessment of the efficiency of the two alternative approaches for several and dynamically changing work scenarios. It is worth noting that our goal was to assess the impact of the protocol on collaborative work. Thus, our model completely ignores the efficiency of the implementation and the load of the network. Moreover, reliability of links and peers is not modeled: instead we just consider peers as *on-line* or *off-line* with respect to the Internet.

A critical issue concerns the detail level of the models, that when performing analysis must be carefully chosen in order to avoid state space explosion. The formalism adopted for the modeling and analysis is a *colored* flavour of stochastic Petri nets called Stochastic Well-formed Nets (SWNs) [4]. Both quantitative and qualitative analysis can be performed on these models. The peculiar feature of SWNs is that due to the particular syntax adopted for the model color structure behavioral symmetries can be automatically discovered and exploited to build an aggregate state space and corresponding stochastic process (a lumped Continuous Time Markov Chain).

The paper is organized as follows: In Section 2 the SWNs formalism is introduced; in Section 3 the models of the client-server and peer-to-peer protocols for collaborative work are presented; some methodological issues involving the adopted modelling approach are discussed; in Section 4 we present and comment on performance indices derived from the model steady-state analysis: these indices allow us to assess the impact of the two alternative solutions on cooperative work scenarios characterized by small group of developers; the complexity of the model analysis, both in terms of memory (state-space explosion) and time, is also addressed. Finally, in Section 5 we summarize the main results achieved and we outline possible directions for future work.

2 Stochastic Well-formed Nets

We recall here only the basic notation and concepts about the SWN formalism needed to understand the models presented in the paper. We assume that the reader is familiar with the PN and Generalized Stochastic PN formalisms [3] (GSPN, the unfolded version of SWN), and has some basic knowledge of high level PN (HLPN) extensions [8] (see [4] for a complete definition of SWNs)

As in all HLPN formalisms, tokens in places are associated with an identifier (color), similarly transitions are parameterized, so that different (color) instances of a given transition can be considered for enabling and firing. Arc functions associate each transition instance with a multiset of colored tokens to be withdrawn from/put into a place.

A SWN model is a eleven-tuple

$$\left(P, T, C, \mathcal{C}, W^+, W^-, H, \Phi, \Pi, \Omega, \mathbf{M}_0\right)$$

where P is the set of places, T is the set of transitions, $C = \{C_1, \ldots, C_n\}$ is the set of basic color classes, each basic color class C_i is a finite, non empty set of colors. Each basic color class can be partitioned into static subclasses $C_{i,1}, \ldots, C_{i,k}$, if it is necessary to make a distinction among groups of colors of the class. The splitting of basic classes is necessary to represent asymmetric qualitative and/or quantitative behaviors of components of the same nature [4].

$\mathcal{C}$ is a function associating a *color domain* to each place and transition. A color domain is defined as the Cartesian product of basic color classes (the same color class may appear several times in a color domain), hence the color associated with tokens in place p as well as the *color instances* of a transition t, take the form of *tuples* of basic color class elements.

The color domain of t is implicitly defined by the set $Var(t)$ of *variables* inscribing the arcs (whose kind may be input, output and inhibitor - I, O, H) surrounding t. Each variable (that may be also seen as a *projection* function) refers to a given basic color class C_{x_i}. The color domain $\mathcal{C}(t)$ is defined as $C_{x_1} \times \ldots \times C_{x_m}$, $x_i \in Var(t)$. A color instance of t ((t, c)) can thus be interpreted as a consistent assignment of colors to the variables in $Var(t)$.

W^-, W^+ and H represent respectively the input, output and inhibitor arcs of the model, and the associated arc functions. The function f on an arc connecting place p and transition t is defined as $f : \mathcal{C}(t) \rightarrow Bag(\mathcal{C}(p))$ where $Bag(A)$ denotes the set of all multisets that can be built on set A; the syntax for expressing such functions takes the form of a sum of weighted and guarded tuples of *elementary functions* defined on the basic color classes. The allowed elementary functions are the *the projection*, selecting one element of a transition instance color tuple (projection symbols can be arbitrarily chosen: e.g., in our examples we use symbols $x, y, st, \ldots$), and the *diffusion* constant functions, returning the set of all elements in a given basic color class (S) or in a given static subclass ($S\, C_{i,j}$) [5].

Φ is a T-indexed function, expressing each transition guard: a guard is used to restrict the set of admissible color instances of a transition, to those satisfying a given predicate. The predicate must be expressed as a boolean expression whose basic terms are *standard predicates*, representing simple conditions on the transition instance tuple elements. Standard predicates are defined to test either equality of pairs of colors bound to variables in $Var(t)$ (e.g., $x = (<>)y$), or to check the membership of a color to a specified static subclass (e.g., $d(x) = (\neq)C_{i,k}$).

The transition priority Π is a T indexed function associating a priority level (in $\mathbb{N}$) to each transition; priority level 0 is reserved for *timed* transitions (graphically represented as white boxes), while all other priority levels are for *immediate* transitions (graphically represented as black bars), which

[4] A basic color class may be circularly ordered, but for simplicity we omit this part of the definition here, being not used in our models.

[5] The successor of a projection is also allowed on ordered basic color classes.

fire in zero time; conflicts between transitions with different priority level are deterministically resolved in favor of the transition with higher priority. Finally Ω is a T indexed function assigning a firing rate to each timed transition t ($\Omega(t)$ is the rate of the exponential pdf characterizing the random firing delay associated with t) and a *weight* to each immediate transition, used to probabilistically characterize the conflict resolution policy between immediate transitions with equal priority.

A transition instance (t, c) has concession in marking $\mathbf{M}$ iff

(i) for each input place p of t $W^-(t,p)(c) \leq \mathbf{M}(p)$ $\wedge$

(ii) for each inhibitor place p of t $H(t,p)(c) > \mathbf{M}(p)$ $\wedge$

(iii) $\Phi(t)(c) = \text{true}$;

(the $>, \leq$ relations and the $+, -$ operations are implicitly extended to multisets).

A transition instance (t, c) is enabled in marking $\mathbf{M}$ if it has concession in $\mathbf{M}$ and no other transition instance with higher priority has concession.

A transition instance which is enabled in marking $\mathbf{M}$ can fire, leading to the new marking $\mathbf{M}'$:

$$\forall p \in P, \mathbf{M}'(p) = \mathbf{M}(p) + W^+(t,p)(c) - W^-(t,p)(c)$$

Here we only recall that as a result of the adopted time representation, the reduced Reachability Graph of a SWN (the graph obtained by suitably removing the *vanishing* markings, i.e., those markings enabling at least one immediate transition) is isomorphic to a Continuous Time Markov Chain (CTMC) (see [4] for a detailed discussion).

The peculiar characteristic of SWNs is that (due to the particular syntax adopted for color domains, arc functions, and guards) behavioral symmetries can be automatically discovered and exploited to build an aggregate state space (called *symbolic reachability graph* or SRG) [5] and corresponding stochastic process (a *lumped* CTMC). However in order to fully characterize the underlying stochastic process, the immediate transitions priority and weight specification must be properly specified by the modeler (see [3] for a discussion of this topic concerning GSPNs).

3 The SWN models of two alternative architectures

In this section we present the SWN models of the client-server architecture adopted by consolidated configuration management tools, and the (new) peer-to-peer architecture implemented by the `PeerVerSy` tool. The models have been built to estimate the impact of the two approaches on small groups of developers freely cooperating, and intermittently connected to some network infrastructure (a LAN, the Internet, etc.). The starting point for producing models has been the `PeerVerSy` architecture specification [1], a mix of pseudo-

code and natural language, integrated with some UML diagrams.

The SWN models have been developed and analyzed (by means of the `GreatSPN` tool [6]) by the following steps: (1) first the model of the client-server architecture was built; (2) the (complex) model of the peer-to-peer architecture was defined by adopting a consolidated compositional approach, based on place superposition, supported by the GreatSPN tool [2]; (3) a submodel was refined to represent the infrastructure that realizes the distributed repository of artifacts; (4) since submodels (and the overall models as well) are fully parametric, we made explicit the relations among component parameters. The main model parameters are:

- the number of cooperating workers,
- the number of artifacts on which workers collaborate,
- the association between a set of artifacts and one worker who plays the role of authority for them.

Parameters are instantiated by properly defining the model *color domains* and *the initial marking*.

Throughout our work (e.g., in step (2)) we tried to establish a relation between different abstraction levels: a thorough discussion on how *behavior inheritance* notions can be used to establish consistency relationships between (sub)models at different abstraction levels may be found in [14] (by which we have been inspired).

Concerning the abstraction level of models, we observe that very detailed models (although may be very useful for integrating and completing the existing documentation) are hardly treatable for performance analysis because of the state-space explosion they trigger.

The following issues have been considered in choosing a convenient abstraction level for our models: (I) whether to represent the underlying communication infrastructure and resource contention (for CPU usage and communication links), (II) whether to represent a single artifact or a set of artifacts shared by the group of developers (the former situation being better supported by the client-server architecture, while the latter one by the peer-to-peer architecture), (III) whether to consider possible failures of the communication links or of the nodes of the network (obviously the client-server architecture presents a single point of failure, while the peer-to-peer architecture is intrinsically more fault-tolerant).

In this paper we decided to not consider neither the communication infrastructure nor the possible occurrence of faults. We focused indeed on the precise description of the client/peer life cycle. Moreover, we restrict our analyses to collaboration based on a single artifact. These simplifications on one side allowed us to study the performance of many more configurations than those manageable on the detailed models by the `GreatSPN` tool. On the other side, coherently with the final goal of the paper (i.e., providing a first assessment of the impact of the the peer-to-peer protocol on the cooperative work),

they correspond to a worst-case assumption for the peer-to-peer architecture.

Another motivation for studying the system behavior with explicit representation of resource contention and/or in presence of faults separately, stems from the observation that the specification of related stochastic parameters (differing for several magnitude orders from the parameters characterizing user activities such as check-in and check-out) should cause stiffness in the underlying stochastic process.

Basic Color classes

Both SWN models have two basic colour classes: UID, denoting the group of cooperating workers, and ST denoting the possible status (*online,offline*) of a worker (ST is accordingly partitioned in two static subclasses *on,off* each of cardinality one). The cardinality of class UID is one of model's parameters. In addition, the SWN model of the peer-to-peer configuration tool has the basic class UPD, which denotes the status (*updated,non-updated*) (corresponding to the UPD's static subclasses *upd,noupd*) of the local copy of the artifact owned by a given worker. [6]

3.1 The SWN model of a traditional versioning system

The SWN model of the client-server architecture for a configuration management tool is depicted in Figure 1. In both models, timed transitions represent time-consuming activities (time being spent for decision and/or execution), while immediate transitions represent *logical* (sequences of) actions. On-line and off-line periods of a client/peer are also represented by means of a timed transition.

Color domains of the model

Let us briefly comment on the definition and the interpretation of the place and transitions color domains (the discussion made here applies to a consistent part of the peer-to-peer architecture's model). All places of the net but place *Status* have color domain UID: a token $\langle u_1 \rangle$ in one of these places represents that worker u_1 has reached a particular status of his/her life-cycle.

In the real world, each client checks out a working copy with a specific version number, and by comparing this with the version of the last copy of the artifact on the server he is able to establish the up-to-date status. In order to exploit the symmetries of the model, we chose not to use the version numbers but to directly maintain the up-to-date status information in the place *uptodate*.

Place *Status* has color domain $\mathcal{C}(Status) = UID \times ST$: a token $\langle u_1, on \rangle$ in *Status*, for example, means that u_1 is currently on-line.

[6] In the client-server model the up-to-date status of the artifact is denoted by the marking of the homonym place

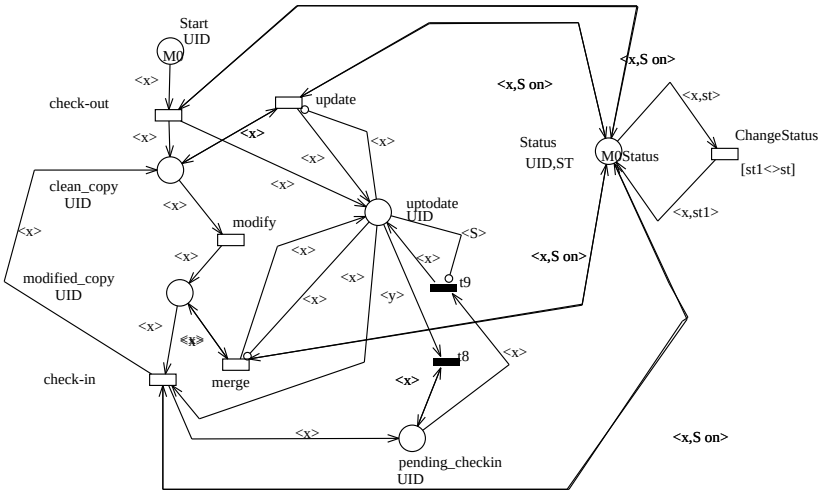


Fig. 1. The SWN model of a server-based version system

The set of model's timed transitions has color domain UID: a color instance of any transition in this group represents a particular action performed by a given worker during his/her life-cycle. These actions may be accomplished by *on-line* users only, because they require the access to the server. The only notable exception is the $modify$ action that can be accomplished off-line on a previous checked-out version of the artifact.

Timed transition $ChangeStatus$ has color domain $\mathcal{C}(ChangeStatus) = UID \times ST \times ST$: a color instance $\langle u_1, s1, s2 \rangle$ means that the current connection status ($s1$) of a worker (u_1) changes (the new status being $s2$) with a given frequency, that is one of the timing parameters of the model. In other words, transition $ChangeStatus$ models the *online* attitude of the workers.

The two immediate transitions $\{t8, t9\}$ (having color domains $UID \times UID$ and UID, respectively) permit us to keep coherent and correct the up-to-date status information after the check-in operations. In particular the cumulative effect of the firings of such transitions as consequence of a check-in is to empty the *uptodate* place except for the token with the user-id (UID) of the client that accomplished the check-in action.

3.2 The SWN model of a peer-to-peer version system

The SWN model of the peer-to-peer architecture for a configuration management tool is depicted in Figure 2. It comprises two main parts, each enclosed in the picture within a dashed box: i) the representation of the main peer life-cycle, obtained by simply refining the SWN model in Figure 1, ii) the representation of the infrastructure reaction to a peer that goes online.

With respect to the client-server architecture's model, places *Repository*

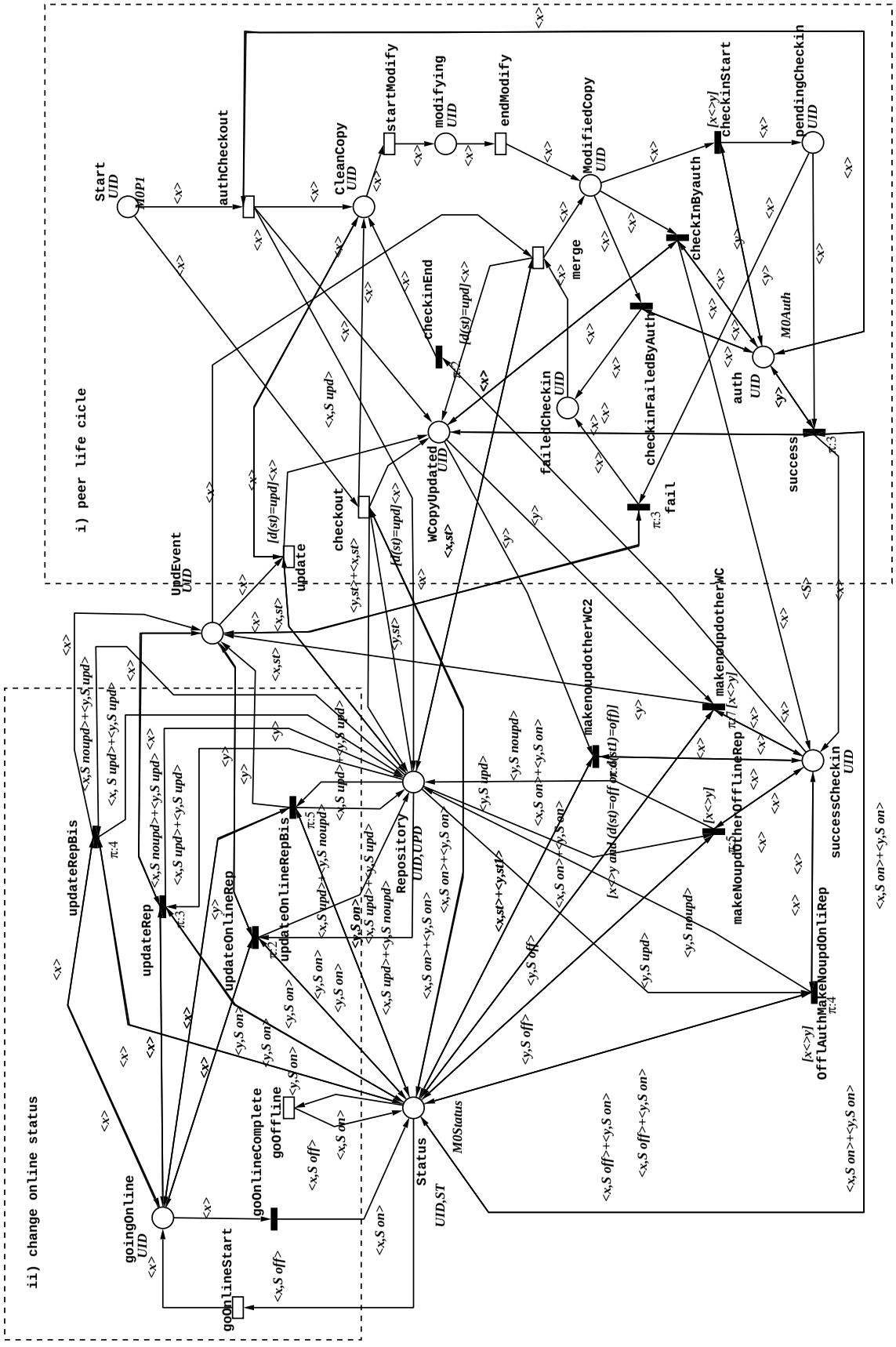


Fig. 2. The SWN model of a peer-to-peer version system

and $WCopyUpdated$ extend the *uptodate* place considering not only the status of the explicitly checked out copy ($WCopyUpdated$), but also the status of the repository replica automatically managed by the `PeerVerSy` infrastructure. The *Repository* place has color domain $UID \times UPD$ (a new color domain): a token $\langle u_1, noupd \rangle$ in place *Repository* means that the *working copy* owned by u_1 is not updated.

Finally another place ($UpdEvent$) has been added in order to represent the fact that the system sends a signal to a peer that a new version of the artifact is available. We highlight here that this is an important peculiarity of the `PeerVerSy` system: it makes aware the peer as soon as possible that the working copy is not still up-to-date. Using such an information, the peers are sometime able to avoid modify actions on old versions of the artifact.

The updating of these three places is managed automatically by the infrastructure. In particular the upper part of the net represents what happens when a peer goes online. In the case that the version of such a peer is newer than those of the already on-line peers, the repository replica of such peers are automatically updated and a "New version" signal is sent to them. On the contrary in the case that the peer copy is older, its repository is updated and he receives the signal. This signal is thus an indication that the repository replica contains a more recent version of the checked out working copy. The bottom part of the net represents what happens when a check-in is accomplished: the working copy status of all the other peers is set to non-uptodate. The repository replica status of all the off-line peers is set to non-uptodate. A new version available signal is sent to all the online peers. All the conflicting pending check-in are then aborted ($fail$ transition). Finally is also considered the case that the check-in is accomplished by the authority when he/she is off-line. In this case no signal is sent and all working copies and repository replica are set as non-uptodate.

Regarding the peer life-cycle (sub)model, we had to differentiate the authority from normal peers in some operation because the prerequisites are different (e.g., the authority can accomplish a check-out, or a succeeding check-in, also if he/she is off-line). However the main change is in the check-in action. In fact this becomes a complex operation that is executed in several steps. In particular the peer can issue a check-in request also if he and/or the authority are not currently on-line. The status of such check-in is thus *pending*, waiting for a condition in which it is possible to evaluate whether the check-in succeeds or not. Several conflicting check-in can be at the same moment in the pending status.

4 Peer-to-peer against client-server: preliminary evaluation

Generally speaking, performance analysis of the SWN models can be used for several purposes: (1) for model validation, in order to compare performance

measures of models at different detail of abstraction: in this way the approximation introduced by abstraction can be evaluated. (2) When comparing the two alternative architectures, to have a first hint on the impact on the reference scenario (small groups of users cooperating on the same document) of the peer-to-peer protocol: the difference between the results corresponding to alternative models allows one to derive some preliminary conclusions on the merits and problems of each, even if the absolute measures are not very precise. (3) In the model's tuning phase, to achieve more accurate and reliable performance indices: all approximations introduced in the modeling phase should be carefully considered, and a validation of the model against actual measures on prototypes should be performed. The results presented in this section are examples of second and third model usage scenarios.

4.1 Managing the model complexity

Before commenting on the obtained performance figures (selected among a number), let us describe the techniques we employed in order to manage the complexity of the analysis performed on the SWN models presented in the section above (in particular, of the peer-to-peer protocol model).

Exploitation of symmetries

The models we are considering in this study are *highly symmetric*, the symmetry of a SWN model being measured by the splitting level of basic color classes into static subclasses. We do appreciate that both models we have developed are fully symmetric with respect the color class UID, which means a *potential* reduction factor (with respect to the number of nodes of the ordinary model's RG) of $n!$, where n is the cardinality of UID.

In order to exploit the behavioral symmetries present in the models, however, we need to define a suitable initial *symbolic marking* (SM). The SM notion makes it possible to define an equivalence relation between states in a purely syntactical fashion: two ordinary markings are equivalent if they can be obtained from each other by applying a permutation on the objects of split color classes that preserves static subclasses. The SWN symmetry property ensures that equivalent firing sequences can fire from equivalent markings.

Referring to the model in Fig.1 the formal representation of its symbolic initial marking $\widehat{\mathbf{M}_0}$ is:

$$\widehat{\mathbf{M}_0}(Start) = \langle Z_{UID} \rangle;\ \widehat{\mathbf{M}_0}(Status) = \langle Z_{UID}, Z_{on} \rangle$$
$$Z_{UID} = n, Z_{on} = 1.$$

The symbolic marking representation is given in terms of *dynamic subclasses* (Z_{UID}, Z_{on} in the example). Dynamic subclasses define a *parametric partition* of static subclasses. Every dynamic subclass refers to a static subclass or to a whole basic class (in the event of non split class), and has an associated cardinality. The symbolic marking above represents the system configuration where all workers are in place (state) $Start$, and they are all

on line. Note that it corresponds to exactly one ordinary marking (this is a frequent situation for the initial SM). A Symbolic Firing Rule allows the building of the Symbolic Reachability Graph (SRG) of the model, working directly at the level of the SM representation.

The formal representation of the symbolic initial marking $\widehat{\mathbf{M}_0}'$ of the model in Fig.2 is a bit more complex:

$$\widehat{\mathbf{M}_0}'(Start) = \langle Z^1_{UID} + Z^2_{UID} \rangle; \widehat{\mathbf{M}_0}'(Status) = \langle Z^1_{UID} + Z^2_{UID}, Z_{on} \rangle; \widehat{\mathbf{M}_0}'(Auth) = \langle Z^1_{UID} \rangle$$
$$Z^1_{UID} = 1, Z^2_{UID} = n - 1, Z_{on} = 1.$$

The marking $\widehat{\mathbf{M}_0}'$ represents a generic system configuration similar to the one described by $\widehat{\mathbf{M}_0}$ for the other model: it means in addition that one (arbitrary) user is chosen as the authority for the artifact. The identity of the authority is not fixed, so that $\widehat{\mathbf{M}_0}'$ actually corresponds to a class of ordinary markings. Since there are n different (equivalent) possible choices, we argue that in this SWN model the *actual* reduction factor due to the symmetry exploitation should be divided by n, i.e., it will be at most $(n-1)!$.

Reducing the number of vanishing markings

Symmetry exploitation alone is not sufficient to make the analysis of the model in Fig.2 manageable: due to the enormous number of vanishing SMs generated during the model's SRG construction (because of the frequent interleaving among immediate transition firings), only configurations with very small groups of users (< 4) can be actually analyzed. Structural analysis techniques may significantly improve the effectiveness of state-space based analysis. In [7] the basis for a *symbolic* approach to express structural relations in *coloured* nets is proposed. The main contributions provided in [7] are the possibility to represent structural relationships between transitions, such as conflict or causal connection, by means of algebraic manipulation of the colour annotations of the net. They show how the theory may be practically applied to the class of (colored) Unary PN to compute *extended conflict sets* (ECS): ECS (originally defined in [3] for GSPNs) represent sets of immediate transitions that are not independent, i.e., such that the firing of one transition in the set may affect the enabling status or the firing probability of the other transitions in the same set. This step on one side is essential to the correct net-level definition of the stochastic process representing the behaviour in time of stochastic Petri nets with immediate transitions (see [13]). On the other side, it allows a drastic reduction of the number of vanishing markings generated during the construction of the reachability graph: transitions belonging to different ECS may indeed fire according to an arbitrary order (fixed, for instance, by assigning different priority levels to different ECS), without altering the model (quantitative/qualitative) semantics. When considering high level stochastic Petri nets (such as SWN), the dependency relation should be

stated in a symbolic form, to characterize in a compact and parametric way the *color instances* of a given (immediate) transition that can influence the enabling/firing probability of other transition color instances.

The limit of the results presented in [7] is the restricted class of HLPN that is treated. The structural analysis technique presented in [7] has been recently extended to SWNs [10]. Such extension is based on a rather complex formal treatment of the color annotations (arc functions and transition guards) appearing on the high level model: the formal aspects of the calculus is completely worked out in [12].

Let us only better explain the meaning of a structural symbolic relation, by giving the formula describing the structural conflict relation between transitions t and t' when they share an input place p:

$$\mathrm{SC}(t,t') = \overline{W^-(t,p) - W^+(t,p)}^{t} \circ W^-(t',p)$$

The first member $W^-(t,p) - W^+(t,p)$ is the functional description of the multiset of tokens actually removed from place p by the firing of an istance of t. The *support* of such expression $\overline{W^-(t,p) - W^+(t,p)}$ maps the multiset to which it is applied into the set taking those colours whose multiplicity is strictly greater than zero. The *transpose* of $\overline{W^-(t,p) - W^+(t,p)}$ is a functional description of the colours instances of t that actually removes tokens from p. Concluding, applying $\overline{W^-(t,p) - W^+(t,p)}^{t}$ to the set $\overline{W^-(t',p)}$ then the colour instances of t removing from p some token for which t' has need, are obtained.

In a similar manner it is possible to derive the functional description of the other structural relations (indirect conflict, causal connection, mutual exclusion) on which the ECS computation relies. By applying this structural analysis technique on the model in Fig.2 we were able to partition the set of immediate transitions into a number of independent ECS with different priority levels, so making the analysis of larger configurations of users feasible.

Removing transient markings

Another simple way for reducing (even if with a lower effectiveness than the two techniques above) the state-space growth of the SWN model in Fig.2 stems from the particular structure of the model's SRG. We can easily argue that this SRG (for any n) is not *strongly-connected*, however it contains *one maximal* strongly-connected component. A steady-state solution does exist, where transient states (i.e. those SMs, as the initial one, that are not home-states) have an associated null probability. We can thus skip xthe transient part of the SRG, obtaining an equivalent (in terms of stationary probability vector) stochastic process whose states have a non-null stationary probability. This actually corresponds to skipping the checkout actions, and it is achieved, for instance, by setting as initial SM of the SWN model $\widehat{\mathbf{M}_0}''$:

$$\widehat{\mathbf{M}_0}''(CleanCopy) = \langle Z^1_{UID} + Z^2_{UID} \rangle;\ \widehat{\mathbf{M}_0}''(Status) = \langle Z^1_{UID} + Z^2_{UID}, Z_{on} \rangle;\ \widehat{\mathbf{M}_0}''(Auth) = \langle Z^1_{UID} \rangle;$$

$$\widehat{\mathbf{M}_0}''(Repository) = \widehat{\mathbf{M}_0}''(Wcopy) = \langle Z^1_{UID} + Z^2_{UID}, Z_{upd} \rangle$$
$$Z^1_{UID} = 1, Z^2_{UID} = n - 1, Z_{on} = 1, Z_{upd} = 1.$$

The simplification just described is also justified by the *transient* analysis made on the original model (that with initial marking $\widehat{\mathbf{M}_0}'$) by means of the `GreatSPN` tools: the state probabilities computed for small values of the time variable indeed shown a rapid convergence of the stochastic process towards its stationary state.

The reduction steps sketched above allowed us to extend the analysis of the peer-to-peer protocol to larger groups of workers. Table 1 shows the sizes of the SRG (in terms of number of nodes) for the two models we developed, faced to the corresponding RG sizes (divided by n). We do appreciate that the reduction factor provided by the SRG is higher for the client-server model than for the peer-to-peer model: this may be intuitively explained by the need of distinguishing the authority-peer from the normal peers. The analysis required 0.250s of CPU time with $n = 2$ and 310m with $n07$ on a Intel Xeon 2.4GHz. The data files generated by the GreatSPN tool with 8 members exceeded the maximum file size (4GB) set on our machine.

Table 1
SRG vs RG

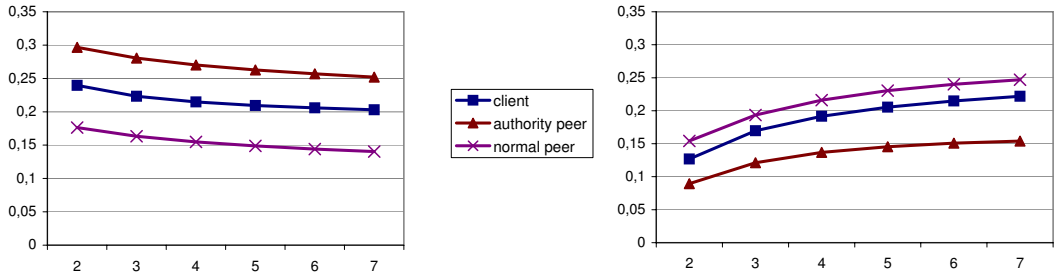
	client-server		peer-to-peer	
♯	SRG (tang. + van.)	RG (tang. + van.)	SRG (tang. + van.)	RG (tang. + van.)
2	63+33	119+66	115+302	230+604
3	399+270	2023+1521	1392+5429	8007+32049
4	1890+1515	31871+30012	10696+55297	223756+1235224
5	7287+6645	478807+543435	61365+394070	5535385+39720045
6	24136+24417	6998159+9326538	286014+2192326	126941010+1141070760
7	71093+78393	100630663+154400169	1139128+10153639	2767706963+30425540229

4.2 Performance figures

In Fig. 3 some results are shown about the performance of the client-server configuration against the alternative peer-to-peer configuration. The plotted curves refer to the average firing frequency (*throughput*) of *checkin* and *merge* transitions, as a function of the number of cooperative workers (up to seven peers/clients are considered), divided by the number of workers.

The *merge* frequency is a metrics that very closely reflects the additional effort spent for recovery actions (whose quantitative evaluation being one of main goals of our analysis), that is introduced by the lack of synchronization typical of optimistic cooperative work scenarios. The *checkin* frequency instead is related to the advances in the project development.

The curves here plotted refer to a "default" setting of timing parameters (the same in both models), and have been chosen as representatives of a *possible* general trend, that however should be confirmed by a more complete



check-in merge

Fig. 3. Mean frequency of checkin and merge in the two alternative configurations

analysis.

Let us remark (and briefly comment on) the simplificative assumptions that were made in setting the model's timing parameters:

- all timed transitions are assigned the same firing delay, except for transition $modify$ that (according to the actual complexity of the involved activities) is assigned a double firing delay.

- any peer/client goes on-line and off-line with the same average frequency (in other words: on-line and off-line periods are assumed to have the same weight).

- we don't differentiate the authority's "work attitude" from that of normal peers: this is (once again) a worst-case assumption for the peer-to-peer model, since in a realistic scenario the authority is the main responsible of the artifact and the one that more often modifies it.

Even if the simplifications above should be carefully considered, the qualitative trend outlined by the curves in Fig. 3 seems reliable. The main highlighted outcome is that (generally speaking) the peer-to-peer protocol behaves in a satisfactory way, comparable to the server based solution. As expected, the authority has a great advantage from the peer-to-peer solution. However, other peers do not pay a big price for this. We experienced that changing some stochastic parameters of the models (e.g., the rate of $modify$ transition) leads to significantly different absolute values, but same curves shape.

5 Conclusion and future work

In this paper we have reported some preliminary results of modeling activity concerning a new peer-to-peer architecture for configuration management tools. A *colored* flavor of stochastic Petri Nets (SWNs) has been used, able to exploit the model symmetries during the analysis phase, thanks to its structured syntax. Two SWN models have been presented: the model of the new peer-to-peer architecture (called `PeerVerSy`), and the model of a traditional client-server architecture. The contribution of the modeling activity has been twofold: on one hand models can be considered as an integration to the avail-

able documentation (specially as concerns the `PeerVerSy` tool); on the other hand, they provide a first hint on the impact of the peer-to-peer protocol on the reference cooperative work scenario (small groups of users cooperating on the same document). Steady state analysis has been performed: a pair of performance metrics characterizing the cooperative work have been computed and discussed. The obtained results about the peer-to-peer configuration are promising and deserve further investigations.

Some methodological issues have been also considered: in particular, we focused on the techniques adopted for managing the model state-space explosion, and on the abstractions used both in the modeling phase and in the setting of stochastic model's parameters.

Besides on extending and completing the performance analysis here presented, we are currently working on a better characterization of the workers behavior by including possible scenarios so far not considered (e.g., the possibility of discarding the work done for example as consequence of a conflicting check-in), and by differentiating class of workers (basically authorities and normal peers). The consequences of a multiple documents (artifacts) scenario are also under investigation.

Acknowledgement

The authors would like to thank Davide Balzarotti, Carlo Ghezzi, and all the anonymous reviewers for their insightful comments on preliminary versions of this paper.

References

[1] Balzarotti, D., C. Ghezzi and M. Monga, *Freeing cooperation from servers tyranny*, in: E. Gregori, L. Cherkasova, G. Cugola, F. Panzieri and G. P. Picco, editors, *Web Engineering and Peer-to-Peer Computing*, Lecture Notes in Computer Science **2376**, Springer-Verlag, 2002 pp. 235–246.

[2] Bernardi, S., S. Donatelli and A. Horváth, *Compositionality in the greatspn tool and its use to the modelling of industrial applications*, Software Tools for Technology Transfer (2001).

[3] Chiola, G., M. Ajmone, G. Balbo and G. Conte, *Generalized stochastic petri nets: A definition at the net level and its implications*, IEEE Transactions on Software Engineering **19** (1993).

[4] Chiola, G., C. Dutheillet, G. Franceschinis and S. Haddad, *Stochastic well-formed coloured nets for symmetric modelling applications*, IEEE Transactions on Computers **42** (1993), pp. 1343–1360.

[5] Chiola, G., C. Dutheillet, G. Franceschinis and S. Haddad, *A symbolic reachability graph for coloured petri nets*, Theoretical Computer Science B (Logic, semantics and theory of programming) **176** (1997), pp. 39–65.

[6] Chiola, G., G. Franceschinis, R. Gaeta and M. Ribaudo, *GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets*, Performance Evaluation (1995).

[7] Dutheillet, C. and S. Haddad, *Conflict sets in colored petri nets*, in: *Proceedings of the 5th International Workshop on Petri Nets and Performance Models*, Toulouse, France, 1993, pp. 76–85.

[8] Jensen, K. and G. Rozenberg, editors, "High-Level Petri nets: Theory and Applications," Springer-Verlag, 1991.

[9] Kung, H. T. and J. T. Robinson, *On optimistic methods for concurrency control*, ACM Transactions on Database Systems (TODS) **6** (1981), pp. 213–226.

[10] L.Capra, M. D. Pierro and G. Franceschinis, *An application example of a symbolic calculus for swn structural relations*, in: *Proceedings of the 7th International Workshop on Discrete Event Systems*, Reims, France, 2004, to appear.

[11] Monga, M., *Supporting nomadic co-workers: an experience with a peer-to-peer configuration management tool*, in: S. Horiguchi, K. Ochimizu and T. Katayama, editors, *Proceedings of the International Symposium on Towards Peta-Bit Ultra-Networks*, IEEE (2003), pp. 79–87, invited paper.

[12] Pierro, M., "Structural analysis of conflicts and causality in GSPN and SWN: theory and applications." Ph.D. thesis, Università di Torino, Turin, Italy (2004).

[13] Teruel, E., G. Franceschinis and M. De Pierro, *Well-defined generalized stochastic petri nets: A net-level method to specify priorities*, IEEE Transactions on Software Engineering **29** (2003), pp. 962–973.

[14] van der Aalst, W. and T. Basten, *Life-cycle inheritance, a petri-net-based approach*, in: *Proceedings of ICATPN 97*, Lecture Notes in Computer Science **1248** (1997).