A The RENEW Version of the Model

We use the Java-based Petri net tool RENEW [11] to execute the scenario. The sources are available at `https://github.com/koehler-bussmeier/bos`. The following nets are used to produce the simulation results. Some minor modifications of the original HORNET model, as explained below, are needed to meet the syntax of the RENEW tool.

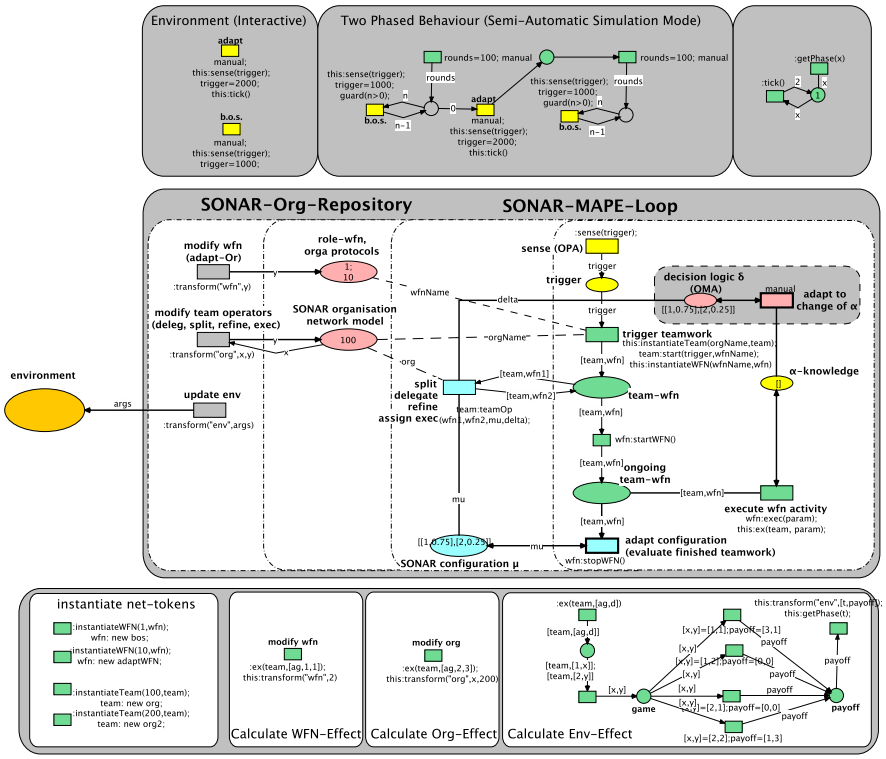

Fig. 5. The SONAR-Mape-Loop

The RENEW Petri net of the SONAR-MAPE-Loop is given in Figure 5. Its main activity, the formation of teams, is started by the transition sense (OPA). It has an empty preset as it is activated over the channel :sense(trigger). This channel is activated from the block Two Phases by the transitions b.o.s. and adapt, which are inscribed by the calling side of the channel, denoted as this:sense(trigger). We have two triggers: trigger=1000, which is a trigger for a 'normal' battle-of-sexes interaction, and trigger=2000, which enables an adaption. The inscription manual denotes that in the interactive simulation these transitions are excluded from automatic firing.

Each trigger enables a *task*. The transition trigger teamwork takes this trigger and generates a team-wfn, i.e., a pair [team,wfn] of a team and a WFN to handle the task that is triggered. The transition has the places role-wfn, orga protocols and SONAR organisation network model as side conditions, which is used as a 'database' to make a look-up. The inscription this:instantiateTeam(orgName,team) takes the orgName and generates a fresh net-token team, which is a new instance of the SONAR-Organisation model as given in Figure 6. At the same time, we synchronise with this net-token with the call team:start(trigger,wfnName), where the team maps the trigger onto wfnName. Then, this:instantiateWFN(wfnName,wfn) maps wfnName onto a fresh net-token instance wfn.

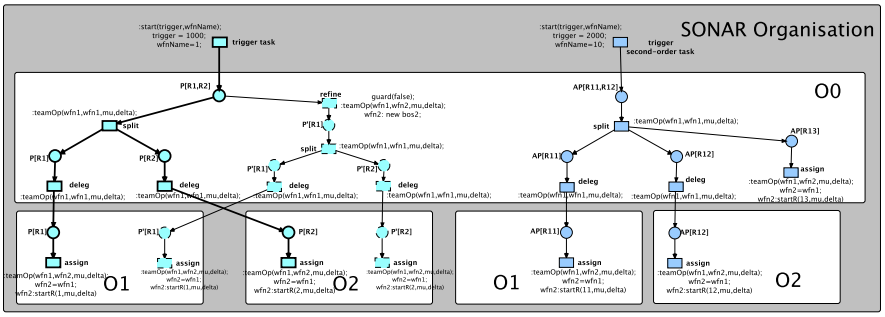

Fig. 6. The SONAR-Organisation (The shaded nodes show the elements that are added by the second-order workflow given in Fig. 8.)

In the initial marking we have exactly one organisation orgName=100 on the place SONAR organisation network model and the two WFN wfnName=1 and wfnName=10 on the place role-wfn, orga protocols. The block titled instantiate net-tokens provides the mapping of the net-tokens (which are used like names) to 'real' net-tokens. For example the top-most transition has the inscription :instantiateWFN(1,wfn); wfn: new bos, i.e. when called for wfnName=1, then we create a new net-token of the type bos as shown in Figure 7; when called for wfnName=10, then we create a new net-token of the type adaptWFN as shown in Figure 8.

When we have generated the pair [team,wfn] the team is a net-token of the organisation as in Figure 6. It has already fired a transition, either trigger task or trigger second-order task, depending on the trigger that has lead to the instantiation of this net.

Let us assume that trigger=1000, then we have synchronised via team: start(trigger = 1000, wfnName). The organisation net in Figure 6 has exactly one transition, namely trigger task, which has the corresponding side of the channel, as specified by the inscription :start(trigger,wfnName); trigger = 1000; wfnName=1. The synchronisation results in a black token on the place P[R1,R2], which specifies the task that is handled by the protocol P with wfnName=1 that has the roles R1,R2. Analogously, the trigger=2000 would fire trigger second-order task on the right.

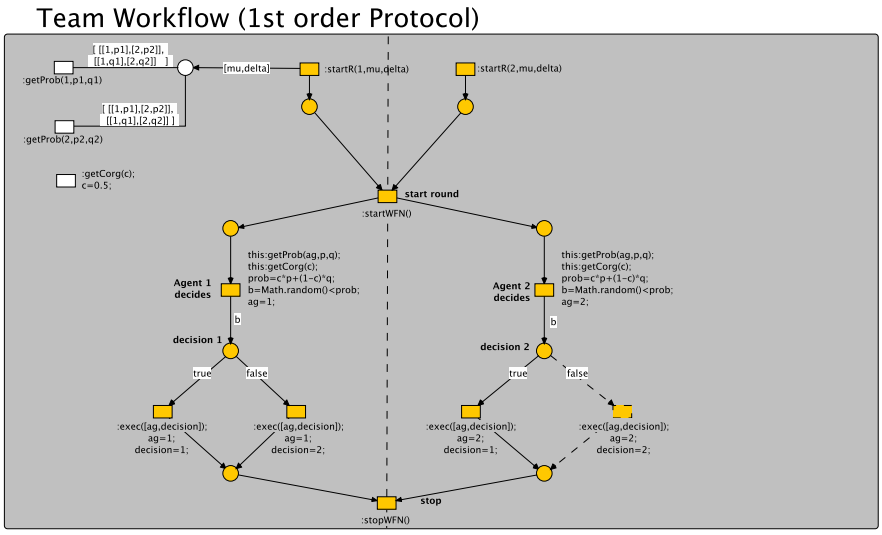

Fig. 7. Stochastic WFN: battle-of-the-sexes (The shaded transition will be deleted by the second-order workflow given in Fig. 8.)

At this moment the team-formation has just begun. The team is generated as an ongoing interaction of the MAPE-loop with the organisation net using the transition deleg, split, refine, exec. This transition synchronises via team:teamOp (wfn1,wfn2,mu,delta) with the team. The team net-token has several transitions labelled with the corresponding synchronisation channels :teamOp (wfn1,wfn2,mu,delta). The inner state of the team decides which of them are enable. Usually, we have more than one enabled transition, which, in general, leads to a combinatorical explosion of possible teams that could be generated by the organisation for a given trigger/task.

Sooner or later, each of these formation processes must end with several firings of transitions named assign, drawn at the bottom of Figure 6. There must be exactly one assignment for each role, i.e., for the battle-of-sexes protocol in Figure 7 we have two roles $R1$ and $R2$ and the team formation must execute two assignments using the channel wfn2:startR(ag,mu,delta), once for ag=1 and once for ag=2.

Here, μ and δ are parameters provided by the MAPE-loop; they are tokens on the places decision logic δ and SONAR configuration μ. Here, the decision logic δ describes the stochastical model of the agents (which are called *organisation member agents* (OMA) in SONAR). The SONAR configuration μ describes the stochastic model that describes how conflicts between transitions with a enabled channel :teamOp (wfn1,wfn2,mu,delta) are resolved during the team formation, i.e. μ models a logic, which concrete team will be chosen for a task with which probability.

Whenever the two roles $R1$ and $R2$ have been assigned in the of Figure 7 the interaction protocol is ready to start as specified by the channel :startWFN(). The calling side is in the center of the MAPE-loop and is inscribed as wfn:startWFN().

The pair [team,wfn] is placed on ongoing team-wfn, where it is located during the whole execution of the WFN until the WFN terminates, which enables the channel wfn:stopWFN() at the transition adapt configuration (evaluate finished teamwork) In between these start and stop events the side transition execute wfn activity fires synchronously with each single action in the WFN. Synchronisation is specified as wfn:exec(param); this generic inscription is used for 'normal' actions as well as for transformation steps as well. The concrete choice of the effect is specified by this:ex(team, param). Depending on the value of param one of the three lower block (Calculate WFN-Effect, Calculate Org-Effect, or Calculate Env-Effect) is used. Each block will inspect the value of param and depending on the values synchronises with one of the three transitions on the left: modify wfn, modify org, or update env. In the case of the block Calculate Env-Effect), which is used whenever the normal battle-of-the sexes protocol of Figure 7 is executed, then we map the two independent choices of the interacting agents and map then onto the game-theoretical payoff as specified in the matrix of the game.

In the WFN of Figure 7 the choice between the two options a and b is resolved by the decision logic δ of the agents and the organisation configuration μ. More concretely, the conflict is resolved randomly by the inscription prob=c*p+(1-c)*q; b=Math.random() < prob; here, p denotes a probability as specified by the organisational μ and q one of the member agents' δ; these two probabilities are combined using the organisation impact parameter c. These values are obtained by the calls this:getProb(task,p,q); this:getCorg(c).

In our simulation we have $p = q$ for both agents. This is encoded in the initial marking [[1,0.75],[2,0.25]] of the place SONAR configuration μ. The place decision logic δ (OMA) contains exactly the same marking. Therefore, we have prob $= p = q$, independently from c. In our simulation the probability for choosing option a is 75% for ag=1 and 25% for ag=2, since this is the optimal mixed-strategy.

Whenever the environment generates trigger=1000, then we synchronised via team:start(trigger=2000,wfnName). The team formation process is quite the same. The organisation net in Figure 6 has exactly one transition, namely trigger second-order task to start a team formation. The main difference is that in this case we have wfnName=10 instead of wfnName=10 and then we create a new net-token of the type adaptWFN as shown in Figure 8. This WFN contains inscritpion of the form :exec([ag,1,1]) or :exec([ag,2,3]). Here the values of the two last two parameters are 'magic numbers' (without any deeper meaning) that are used to synchronise with the appropriate modification transition, i.e., either modify wfn or modify org in Figure 5.

Note, that the order of the transformations is arranged in a way that a new operator in the organisation net is added only after all its predecessors have already been added. Note also, that each transformation is carried out by that agent that is responsible for the new operation, i.e., the agent O_0 executes all add-operations that happen within its own sub-net part of Figure 6, and then the agents O_1 and O_2 concurrently add the elements belonging to their subnets.

At this special point we have to make a little 'trick' as Renew – unlike Hornets – has no inbuilt option to modify the net-tokens structure at run-time. Here, the trick is that we pre-compute the effect of the adaptWFN on the workflow net and to define a new net, called bos2 (cf. Fig. 9) that is exactly the outcome of the

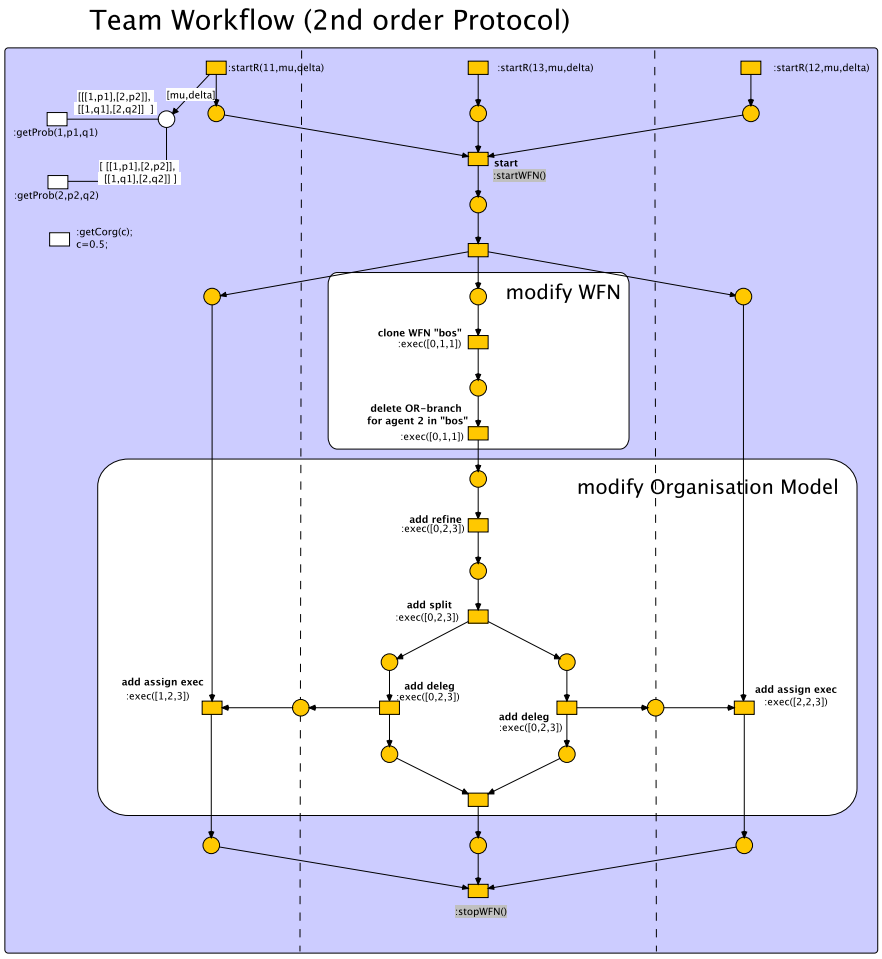

Fig. 8. Second-Order WFN: Team-based Modification of the Workflow Model given in Fig. 7 and the Organisation Model given in Fig. 6

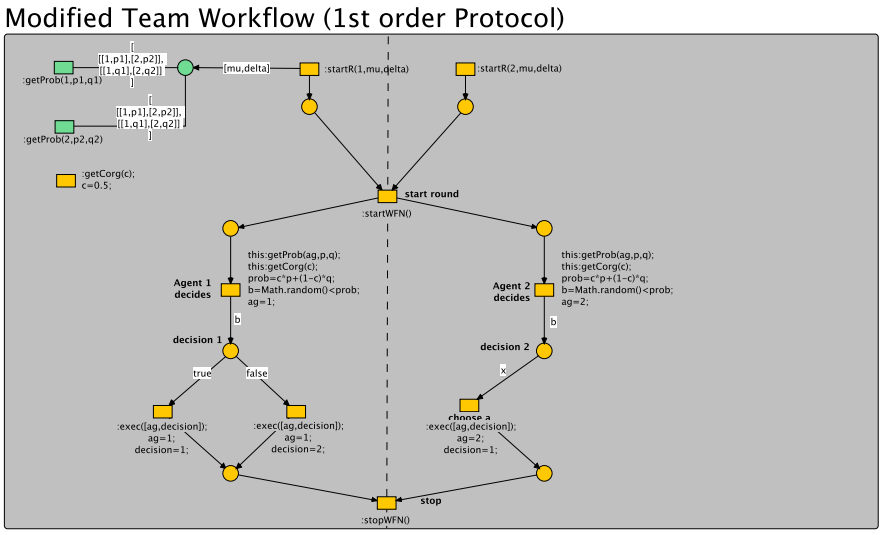

Fig. 9. Pre-Computed WFN that is generated as an Effect of the Second-Order WFN Fig. 8. It is an Modification of the WFN from Fig. 7 and the Organisation Model given in Fig. 6

transformations applied here (i.e. delete the option b for the second agent). The adaption is simulated by the block Calculate WFN-Effect where the transition calls the transformation via this:transform("wfn",2); this has the effect that we obtain a new token wfnName=2 on the place role-wfn, orga protocols and wfn=2 will be mapped onto bos2.

Analogously, we pre-compute the effect on the organisation to replace org=100 by org=200 with the call this:transform("org",x,200). Here, the call this: instantiateTeam(orgName, team) will generate a net-token of the net org2 whenever org=200. The new organisation is shown in Fig. 9. This new organisation now has two possible teams to react on the task, which is specified by the conflict place P[R1,R2].

This conflict is resolved by the decision logic δ of the agents and the organisation configuration μ. More concretely, the conflict is resolved randomly by the same construct that we have already used to resolve the conflict in the battle-of-the-sexes protocol, i.e., by the inscription prob=c*p+(1-c)*q; b=Math.random() < prob; here, p denotes a probability as specified by the organisational μ and q one of the member agents' δ; these two probabilities are combined using the organisation impact parameter c. These values are obtained by the calls this:getProb(task,p,q); this:getCorg(c). To ease the interpretation of our simulation we have p=0.0; q=0.0 in this simulation, i.e. we have prob=0.0, independently from c, which is $c = 0.5$. We form a team as specified by the left branch with probability of prob $= 0\%$ and the team on the recently added right branch with a probability of $(1 - \text{prob}) = 100\%$.

As a special feature of this new team the right branch continues with a refine operation, i.e. the organisation specifies that the task in no longer handled by the

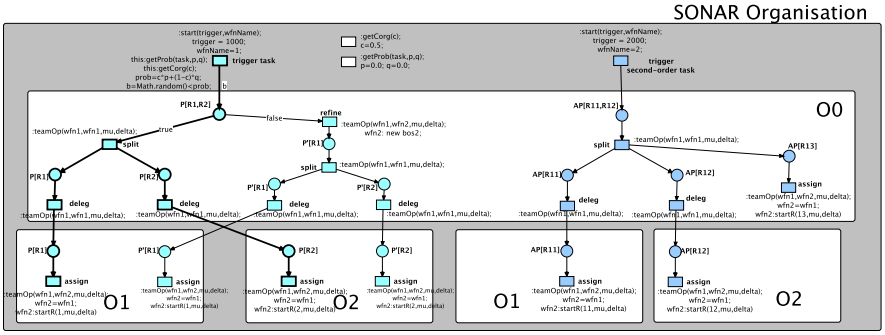

Fig. 10. Pre-Computed Organisation Net that is generated as an Effect of the Second-Order WFN Fig. 8. It is an Modification of the Organisation from Fig. 6

WFN P[R1,R2], but by P'[R1,R2], where the inscription :teamOp(wfn1,wfn2,mu,delta); wfn2: new bos2 specifies that in the MAPE-Loop the current WFN wfn1 is replaced by wfn2: new bos2, i.e. the recently added WFN from Fig. 9.

In general, at the end of an interaction the transition adapt configuration (evaluate finished teamwork) would also perform an update of the SONAR configuration μ, i.e., we allow the organisation to learn from observations. In this small example the value of μ is unchanged during the simulation.

Analogously, each member agent (OMA) will learn, i.e., it will adapt to change and perform an update on the decision logic δ (OMA). In this case study, the agents do not learn.