From EHORNETs to Hybrid Agent and Workflow Systems

Thomas Wagner, Daniel Moldt, and Michael Köhler-Bußmeier

University of Hamburg, Department of Informatics http://www.informatik.uni-hamburg.de/TGI/

Abstract. This paper presents a mapping between an elementary Higher Order Net (eHornet) system and the PAFFIN-system, a practical hybrid agent and workflow system. The EHORNET system provides a formal basis for the processes. It also supports properties, like adaptivity and mobility, which can be preserved in the mapping and used in the PAF-FIN-system. The paper describes the basics of elementary Hornets, the PAFFIN-system, as well as the mapping between them.

Keywords: Workflow processes, Higher order nets, Petri nets, Agent systems, Adaptivity, Mobility

1 Introduction

Using Petri nets and especially workflow nets as a special subset of Petri nets (see [1]) to model business processes is an established and well documented practise. However, workflow nets based on classical Petri nets have a static composition. Employing dynamics or adaptivity in these nets is obstructed by this static nature of the net structure and underlying formalism.

Extensions of coloured Petri nets are one way to improve modelling. However, in order to maintain desirable formal properties of workflows, the conceptual setting of such an extension has to feature a mapping to classical nets. Higher order nets (HORNETS) [12, 13] allow for algebraic operations on nets in a hierarchic setting. This introduces dynamic possibilities for systems designed with HORNETS, while preserving desired workflow properties. However, their formal nature means that some practical aspects are more complicated. Examples for these aspects include distributed execution, which is difficult to achieve in most cases, as well as (modelling of) management aspects, such as resource allocation and access control which require specific practical solutions for the given contexts of the (execution) environment. The adaption of net structure at run-time is usually difficult in practical settings. Some special mapping concepts need to be used to realise this dynamic feature¹.

¹ E.g. in RENEW this is not directly supported, only via tokens being generated at runtime and hence establishing new net structures. Net templates can be constructed at run-time and can be instantiated via the net loader. Doing so we can establish completely new net structures in a similar way as Java does it via reflection and the class loader.

Consequently, the goal of the research presented in this paper is to find a mapping between the formal HORNETs and a practical execution environment. The intended execution environment is a hybrid agent and workflow system called the PAFFIN-system (Processes and Agents for a full Integration, [28]). PAFFINs are execution entities which can dynamically, at runtime, act and be interacted with as agents, workflows, both or something in between. They combine the structural capabilities and properties of agents with the behavioural mechanisms and attributes of workflows. A full mapping between HORNETs and PAFFINs will provide practical execution mechanisms for formal HORNETs as well as a formal basis for the practical PAFFINs. Assumptions and observations made for a HOR-NET model would be viable for the PAFFIN-system. Later on, verification and validation results from the HORNETs may be applied to the PAFFIN-system. This paper presents a coarse approach for the mapping of EHORNETs to PAFFINs for a practical embedding.

The paper is structured as follows: Section 2 discusses HORNETS and describes the example used within this paper, while Section 3 presents the basics of the PAFFIN-system. Section 4 describes the mapping between EHORNETS and PAFFIN-system. Throughout these sections a common illustrative example will be maintained. Finally, Section 5 takes a look at related work, before the paper is concluded with an outlook in Section 6.

2 Elementary Hornets

HORNETS [12, 13] are used for the formalisation. HORNETS are a generalisation of Object Petri Nets, which follow the *nets-within-nets* paradigm [25]. Object Petri Nets are Petri nets, where the tokens are nets again, i.e. there is a nested marking.

For HORNETS Object Petri Nets have been extended with algebraic concepts that allow modifying the structure of the net-tokens as a result of a firing transition. This is a generalisation of the approach of algebraic nets [21], where tokens are algebraic data types. A HORNET is called elementary (EHORNET) whenever there are two levels of nesting, where the outer level is called the *system net* and the inner tokens nets are called *object nets*.

This paper will give a simple example only and refer to [12, 13] for all formal definitions. Consider a HORNET with two workflow nets N_1 and N_2 as tokens – cf. Figure 1. To model a run-time adaption², N_1 and N_2 are combined, resulting in the net $N_3 = (N_1 || N_2)^3$. This modification is modelled by transition t of the HORNETS in Fig. 1. Assume a variable binding α with $x \mapsto N_1$ and $y \mapsto N_2$. This

 $^{^2}$ With run-time adaption we mean that the structure of the net is changed. This means that e.g. a transition is replaced by a subnet while the simulation is executed. In HORNETS this is done by the modification of the tokens that can be constructed at run-time / execution time of the model.

³ In the following we will restrict our examples to the parallel \parallel operator. Other operators introduced in [12] are the operators sequence • and alternative +. For HORNETS the firing rule is defined to be reversible. Therefore, using operators on

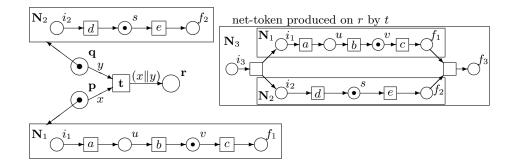


Fig. 1. Modification of the Net-Token's Structure (taken from [12])

binding enables the transition t. Assume that (x||y) evaluates to N_3 . If t fires it removes the two net-tokens from p and q and generates one new net-token on place r. The net-token on r has the structure of N_3 and its marking is obtained as a transfer from the token on v in N_1 and the token on s in N_2 into N_3 . This transfer is possible since all the places of N_1 and N_2 are also places in N_3 and tokens can be transferred in the obvious way.

Workflow nets as introduced by van der Aalst (see [1]) are used as the background for the token structure within this paper. However, the notion of workflow as we use it here in an informal meaning is a generalisation of behavioural descriptions of any entity, not just the formal control of business processes. This allows us to argue in the following example to have persons being described as workflows. In this way we concentrate on the modelling of the behaviour of people and not on the people themselves. In addition to this the workflow nets in the following also need some additional concepts for synchronisation with other workflows for interactions (In the same way as people interact we will talk about interaction / synchronisation of workflows). Mobility of workflows also needs to be covered in the case that the workflow can influence the movement within an environment what is natural behaviour of people. Doing so we are able to model the relevant features which are covered by the EHORNET formalism.

In the following, a scenario featuring mobile and adaptive workflows is discussed. This scenario is used in the remainder of this paper to illustrate the different models. It is an artificial family situation:

A child wants some candy and asks its parents for permission. The parents are in different rooms of the house and the child always asks its mother. If the mother approves the candy consumption the child can eat some

incoming or outgoing arcs works in similar ways. Their informal behaviour can be described as follows: The operator binds the variables to the nets (tokens) on the incoming places and the outgoing places. The usual non determinism applies if there are several tokens within one place.

candy. If the mother disagrees the child becomes sad and won't eat any candy. However, the child can also decide to ask its father, who always agrees to eat all of the available candy in our artificial setting.

This simple scenario can be easily captured as a workflow system. Following our idea that the behaviours of people are modelled as workflows, the child and its parents are modelled as individual, interacting workflows. When the child asks one of its parents the two corresponding workflows need to interact. The child moves between the rooms to interact with its parents.

The scenario features adaptive and mobile behaviour. The child adapts when it decides that it can ask the father after the mother didn't allow eating candy. The child is also mobile as it has to move between the rooms in which its mother and father are located.

The scenario is now presented as an EHORNET system. The three different (workflow) nets (Child, Mother, Father) are shown in Figure 3. They are executed in the system net depicted in Figure 2.

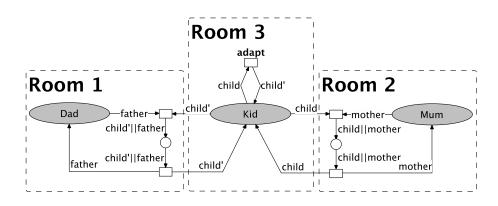


Fig. 2. Overall System Net

The partition of the system net into the three rooms serves to illustrate the movement of the family members during the scenario. The father net instance (the token named "Dad") is in room 1, the mother net instance (the token named "Mum") is in room 2. Initially, the child net instance (the token named "Kid") is in its own room 3 but moves to the other rooms during the scenario⁴.

⁴ The nets are just an informal representation and do not include the inscriptions for the formal representations of the EHORNETS as introduced in [12]. For example for Room 2 somehow the information about the child and the mother net should be kept on a place. The assignment corresponding to net types is also not shown. The yes/no inscription can easily be replaced by a net that hat two transitions. One transition could be relevant for the yes and one for the no answer. The same holds for several other modelling patterns in the example.

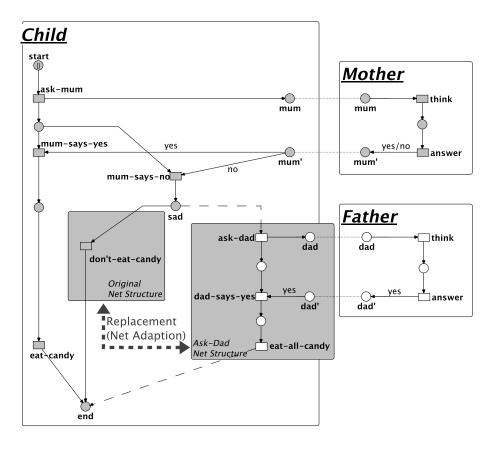


Fig. 3. Child, Mother, and Father Nets

System The system net embeds the overall execution in the scenario. When the child net asks the mother for candy it moves from room 3 to room 2. There it merges with the Mum net, which represents the interaction between the two. When the interaction is finished the nets are separated again and the child returns to room 3. Depending on the outcome of the interaction with the mother, the child workflow may be completed in one of three ways. If the mother agreed the child workflow can be completed directly. If the mother didn't agree the child can either accept this and complete its workflow, or it can change. This is modelled through the **adapt** transition.

The adaption mechanism replaces the part of the child net that accepts the mother's decision with another net structure that asks the father for candy. It removes the child net from the place in room 3 and puts the changed net back instantaneously (the change is indicated by *child*')⁵.

⁵ In this paper we use mostly an informal notation for the EHORNETS. The adaption needs to be covered by inscriptions that allow for the changes of the tokens / nets.

The adapted child net can then move to room 1, merge with the father net, and finally return to room 3 when the interaction is finished. Since the father will always allow candy, the child net can complete its own workflow.

Child The child net represents the main workflow. It begins with the task to *ask-mum* (for the candy). That task enables the merger of the mother and child nets. When these nets merge, the places with the same name (mum/mum') are combined. This initialises the mother workflow.

When the mother's answer is put onto the **mum'** place the mother and child nets can separate again. Depending on the mother's answer either the *mumsays-yes* or *mum-says-no* transitions can fire. If the mother allowed the candy the next transition is *eat-candy*, after which the workflow can terminate. If the mother did not allow the candy there are two options.

The original net implements the child accepting the mother's decision. The *dont-eat-candy* transition fires and the workflow can be completed. If the child decides to ask its father the *Original Net Structure* is replaced by the *Ask-Dad* Net Structure. This adaption is controlled and executed on the system net level. When the adaption transition in the system net is fired the child net is, in one firing step, removed from its original place, the net structures exchanged, and finally the adapted net put back onto the original place.

The Ask-Dad Net Structure is similar to asking the mother. The father, however, always agrees to the candy, so there is no negative option. The child asks its father, thus initialising the father workflow, which returns a yes to the child, which can now *eat-all-candy*. Finally, the workflow is then in its end state.

Mother/Father The mother and father nets follow a similar structure. Since the mother and father in the scenario are just secondary (they are object nets) they are not initialised in the usual way and cannot work without being merged with the child net.

Consequently the structure of both nets is very simple. The start place **mum/dad** enables the sequence of a "think" and an "answer" transition. The answer is finally put on the end place **mum'/dad**'.

When the mother or father nets are merged with a child net for interaction, the start and end places are the merging points. This way the answer can be transferred to the child net when the nets are separated again. The separation is not described here, since it follows similar patterns as the merging according to the details of the firing rule described in [12].

3 Execution Environment

This section describes the basics of the PAFFIN-system (Processes and Agents for a full Integration-system), which provides the hybrid agent and workflow

So to adapt in the Figure 2 means that the child, as informally described in Figure 3 will change its structure from the left net part (Original Net Structure) to the right net part (Ask-Dad Net Structure). The formal inscriptions are not shown here.

management utilised later on. More information about the PAFFIN-system can be found in [28, 29].

3.1 General Idea of the PAFFIN System and its Integration

In general, a combination of agents and workflows is not a new idea. There has been a number of research efforts that combined the two concepts on some level (see the discussion on related work in Section 5). However, most of these research efforts only utilised one concept to enhance the other one, e.g., using agents to implement and improve workflow management. Our research related to the PAFFIN-system goes beyond this. The goal is not just to improve one of the concepts. Rather, the goal is to improve **both** concepts, integrate and combine them and also provide **both** of them in a unified way to application modellers. This is what we call a *full* integration of agents and workflows, as opposed to a *partial* integration which enhances only one of the concepts.

The result is a modelling construct which is no longer just a workflow or an agent, but instead an abstract entity which inherits the capabilities / functionalities of both concepts. Entities can dynamically act and be interacted with as an agent, a workflow, both or something in between depending on the current situation and needs at runtime. Such an entity is, in short, called a PAFFIN or PAFFIN-entity. PAFFINS serve as the main modelling construct within the PAFFIN-system and are modelled and, at runtime, managed similarly to agents and workflows. The ideas leading to the PAFFIN-system were presented e.g. in [27, 26, 20].

A full integration of agents and workflows as realised in the PAFFIN-system provides a number of advantages. The advantages of partial integrations, i.e., the enhancements for one of the concepts by incorporating strengths of the other one, are available for modelling in both directions. PAFFINs as agents can utilise workflow properties like atomicity of tasks and simple user interaction. PAFFINs as workflows can utilise agent properties like intelligence and mobility. Furthermore, since PAFFINs can dynamically be agents or workflows, modellers are no longer restrained to using one of the two exclusively. Rather, modellers are free to flexibly adapt the PAFFIN to the needs of specific situations and choose whether agent, workflow or something in between is best suited at that moment.

For example, a PAFFIN acting as a workflow can, during the execution of its tasks, act as an agent receiving and reviewing intermediate results of the task. If the intermediate results indicate errors or other problems, the PAFFIN, as an agent, can autonomously decide to cancel the execution of the task it executes acting as a workflow. The combination and full integration of the concepts agent and workflow on the same abstraction level provides application modellers with a powerful modelling mechanism. That mechanism, the PAFFIN, is highly versatile, flexible and sophisticated. The downside of the power and expressiveness of the PAFFIN mechanisms is that they are more difficult to use efficiently. Still, this disadvantage can be alleviated with supporting modelling tools, guidelines and approaches.

For the purposes of this paper the usability of PAFFINS as enhanced agents and workflows is most important. The EHORNETS are workflows on the secondary level (the object nets). These workflows interact at some points (merging mother/father and child), they make decisions (the decision to adapt), and they are mobile. Interaction, decision making, mobility and more are classical agent properties. The full integration in the hybrid system allows to model these workflows as enriched PAFFIN-entities, adopting agent properties while retaining workflow behaviour. The adaptivity of the workflows can also directly be mapped into agent behaviour as explained in the remainder of the paper.

3.2 Integration through AGENT ACTIVITIES

The core of the integration within the PAFFIN-entities is the so-called AGENT ACTIVITY (AGAC) (see [28] for details). The principle of the AGAC is illustrated in Figure 4. An AGAC is a modelling construct to describe the behaviour of PAFFINS. It represents an abstract activity consisting of a number of agent actions and workflow operations. Agent actions can be categorised into sending a message, receiving a message and performing an internal action. Workflow operations are requesting a workitem and completing or cancelling/aborting the execution of the corresponding activity/workitem.

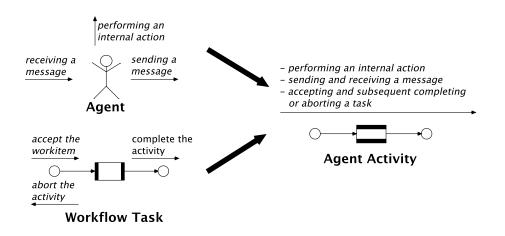


Fig. 4. Principle of the AGENT ACTIVITY (from [28])

Using these three types of operations and three types of actions it is possible to model any agent or workflow behaviour. An AGAC incorporating both actions and operations then describes the abstract activity of a PAFFIN-entity acting as both agent and workflow. Actions and operations are combined in a small process description. Figure 5 shows such a process for an extended task. The net is a reference net [16] that is implemented for the current prototype of the PAFFIN-system (see next subsection). Each action and operation corresponds to one of the "blocks" of net patterns indicated in the net. The following actions are executed:

- 1: Request Workitem Workflow operation This workflow operation describes and offers a workitem for a resource/user within the system. That workitem is part of the PAFFIN executing the net in Figure 5 acting as a workflow. When a user requests this specific workitem the transition can fire and the workitem becomes an activity. Firing also stores data about the activity in the two places which are used to enable completing and cancelling the activity later on.
- 2: Receive Message Agent action The activity in the example requires the resource/user to create some data object and send it as a message to the PAFFIN executing the process in Figure 5^6 . This agent action describes the waiting for and subsequent reception of that message. At this point the PAFFIN acts as an agent while still executing the task acting as a workflow.
- 3: Internal Action Agent action This agent action describes the executing PAFFIN reviewing the received message. It is an internal agent action that produces a boolean result whether the data object created by the resource/user is accepted. The missing third agent action of sending a message is not an explicit part of this extended task example. It is, however, implicitly included in the confirm/cancel in blocks 4 and 5. These management blocks require some communication between the current PAFFIN-entity, the executing PAFFIN-entity and the underlying management of the platform.
- 4: Direct Confirm Activity Workflow operation If the data object created by the resource/user of the activity is accepted by the executing PAFFIN this workflow operation is executed. It implements the executing PAFFIN confirming and completing the activity. At this point (as well as in the alternative described in 5) the PAFFIN is both agent and workflow at the same time. As a workflow it completes the activity (and, automatically in the background, informs the resource/user of the completion) while, as an agent, it autonomously decides to do so. The result of the activity (the created data object) is saved for use in later behaviour and the process can be completed.
- 5: Direct Cancel Activity *Workflow operation* If the data object created by the resource/user of the activity is not accepted by the executing PAFFIN the activity is cancelled here. Again, the executing PAFFIN is both agent and workflow at this point performing the cancelling operation and deciding as an agent to do so. Cancelling the activity resets the workitem by also triggering the lower transition of block 1, reenabling the workitem for another try by another resource/user.

The small and simple example shown in Figure 5 already shows how the combination of agent actions and workflow operations is utilised in AGACS. Through this combination the PAFFINS within the system can maintain a singular

⁶ Note that the resource may itself be executing an AGAC within its own PAFFIN which is not shown here. The implementation of the resource is irrelevant, as along as a compatible message is sent to the PAFFIN executing the net in Figure 5.

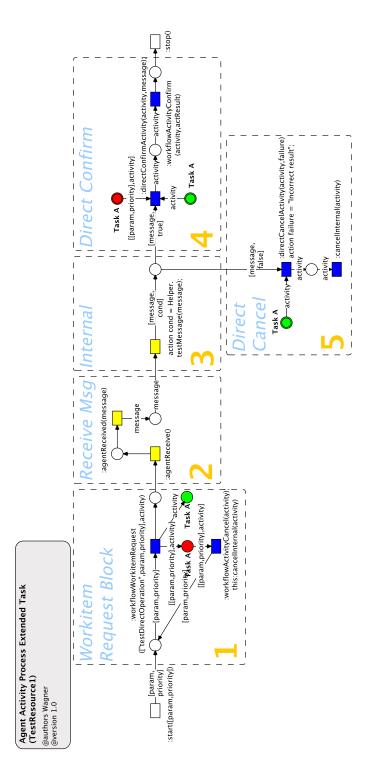


Fig. 5. Process within an AGENT ACTIVITY() (Prototype implementation)

context in which they act as agent (Actions 2 and 3), workflows (Operation 1) or both (Operations 4 and 5). The AGAC is the core mechanism through which the overall integration of agents and workflows within the PAFFIN-system is achieved. More information about the AGAC can be found in [26, 28].

3.3 PAFFIN-System Architecture and Prototype

The prototype of the PAFFIN-system is implemented using reference nets [16] and the RENEW-tool (**Re**ference **NEt** workshop ⁷ (see [4] for a description of the tool). Reference nets are a high-level Petri nets formalism following the nets-within-nets paradigm [24]. Reference nets allow for Java inscriptions on transitions within the net and represent the majority of actually executed code.

Agent aspects of the PAFFIN-prototype are based on the MULAN reference architecture (**MULti Agent Nets** [22]) and its implementation CAPA (Concurrent Agent Platform Architecture [8]). Agent systems in MULAN and CAPA consist of four layers, where each layer contains and serves as the direct execution environment for the next one according to the nets-within-nets paradigm. The system layer contains (i.e. consists of) the agent platforms which contain (i.e. serve as execution environment for) the agents which contain (i.e. execute) the agent behaviour called protocols. Workflow aspects are based on the principles of workflow nets ([1] generally, [11] for reference nets). A workflow net realises the facilitation of a business process as a Petri net. For the purposes of the PAF-FIN-system we consider not just business processes, but any arbitrary process to be modelled as a workflow net.

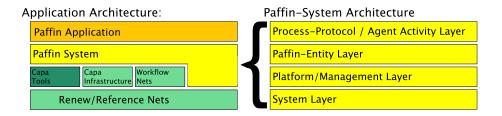


Fig. 6. Application and System Architecture for the PAFFIN-system.

The PAFFIN-system itself adopts the principle four layers of the MULAN reference architecture and extends the layers to capture workflow aspects. This is illustrated on the right-hand side of Figure 6. The *system* level represents the entire system and is a logical summary of the other levels. The system consists of a number of *management platforms*, which serve as the next level. These serve a similar role as workflow and agent management systems in classical system.

 $^{^7}$ RENEW is available at www.renew.de see [17].

The management platforms serve as the execution environment for the PAFFINs, which make up the third level. Finally, the AGENT ACTIVITIES are defined and related to each other in the *process-protocol* level. A process-protocol serves a similar function to an agent protocol or a workflow⁸.

The PAFFIN-architecture describes, as a reference, the internal composition and interrelation between the individual elements of the executed management system. It represents a detailed view of the internal structure. The application architecture shown on the left-hand side of Figure 6 on the other hand describes the structure of an application built with the PAFFIN-system. The foundation is given by RENEW and reference nets. CAPA tools and infrastructure (e.g. monitoring and administration), as well as reference workflow net mechanisms are built on top of RENEW. The PAFFIN-system, whose internal architecture was described above, builds on top of all of these mechanisms as well as RENEW and reference nets directly. On top of the PAFFIN-system the PAFFIN-application is created. The application is what is actually built by a modeller using the PAF-FIN-system. It provides the content of AGENT ACTIVITIES, process-protocols, PAFFIN-knowledge, etc. that actually captures what the modellers want their system to do. The mapping between EHORNETS and PAFFINs creates exactly such an application as will be described in the following section.

The current prototype of the PAFFIN-system already fully supports the full integration as motivated in the previous subsection. PAFFIN-entities of the prototype can execute AGACS that include workflow operations and agent actions in any (reasonable) order. Currently ongoing work on the prototype focusses on supporting the modelling, which addresses the disadvantageous effects of the extended modelling capabilities. Standardised AGAC net components, adapted monitoring and administration tools and simplifications to the modelling process are currently being investigated. Additionally, the applicability of the PAOSE modelling approach (Petri net-based Agent- and Organisation-oriented Software Engineering [3]) for MULAN and CAPA is being examined.

4 Mapping

The preceding sections explored the basics of EHORNET systems and the PAFFIN execution environment. This current section outlines how to map an EHORNET system to a PAFFIN-application. Mapping, in this context, is meant as a rough translation coming from a modelled EHORNET system and yielding a PAFFIN-application containing the same functionality. Figure 7 gives an overview of the mapping process. Mapping can be broken down into two basic steps:

First, identify the structural, behavioural, and additional aspects of the EHORNET system. **Second**, combine these aspects into the appropriate constructs in a PAFFIN-application.

⁸ More accurately a subworkflow of the PAFFIN as a workflow.

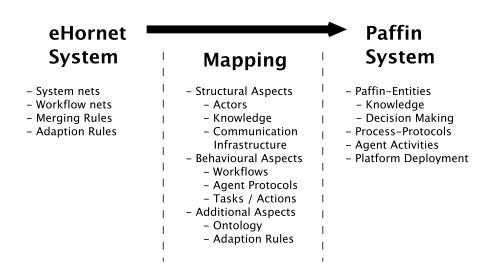


Fig. 7. Mapping between the EHORNET and the Hybrid System

4.1 Identifying the Aspects

The *behavioural aspects* include the processes, the individual tasks/actions, and the access/execution rules. In the EHORNET system, this corresponds to the different workflow nets. In the example introduced in Section 2, the processes are *Child, Mother*, and *Father*. The tasks can be extracted directly from the workflows, as well as the executing resources. In the example the executing resource is always the actor of the same name (e.g. the child executes the tasks of the *Child* workflow).

Structural aspects deal with the individual actors, their properties, and their relationship with one another. The actors themselves need to be identified from the executing resources of the EHORNET workflows. For the example these are the child, the father, and the mother. The communication and infrastructure between the actors is defined in the system net of the EHORNET system. A group of places in the system net will always represent a logical execution environment, while the transitions between these groups define the communication infrastructure. In the example, there are three execution environments, the different rooms. The transitions between the room 3 and both room 1 and 2 represent an infrastructure that allows the child to move from room 3 to room 1 or 2 and back. Consequently, there are three execution environments for the PAFFIN-system, with a communication/migration infrastructure that allows actors from room 3 to move to room 1 or 2 and back. The communication between different actors is modelled by merging and unmerging workflow nets into an EHORNET system. Instances of this merging also need to be identified and analysed for the participating actors. In the example these are the times, at which the child workflow merges with the mother and father workflows to ask permission for candy. The

actors involved are the child and mother in the first case and child and father in the second case.

Additional aspects include those aspects, which cannot be classified into the structural and behavioural groups. These include the ontology, which is orthogonal to the structure and behaviour, and the adaption rules, which affect both structure and behaviour. The ontology needs to be identified from the EHORNET system by analysing which (data) objects are considered and communicated between the actors. For the example this is only the question and answer for candy between the child and one of its parents. To identify the adaption rules, each possible adaption in the EHORNET system needs to be analysed. The circumstances and requirements for each adaption have to be noted, as well as the possible outcomes.

4.2 Combining the Aspects

After having identified the different aspects, they need to be combined to create a PAFFIN-application. The elements which must be modelled for a PAFFINapplication are: entity-knowledge, process-protocols with the included AGACS, decision making of the entities, and the deployment management platforms. The remaining technical administration and management of PAFFINS do not need to be changed for each application.

As a basis, each actor from the *structural aspects* is mapped onto one PAFFINentity each, though there can be multiple instances of an entity (e.g. a second child). These entities are assigned to management platforms according to the identified execution environments from the EHORNET system. The communication/migration infrastructure is coded into the management platforms to allow or disallow communication with certain other management platforms. Communication between PAFFINs is enabled by providing the identifiers and communication information for each communication partner in each PAFFIN's knowledge. In the example, this results in three PAFFINs (child, mother, father) and three management platforms (rooms 1-3). Rooms 1 and 2 manage the father and mother entities respectively and allow only communication/migration to/from room 3. Room 3 manages the child entity and consequently allows communications/migrations to/from both rooms 1 and 2. The father and mother entities know only the identifier of the child entity, which in turn knows both and is capable of migration.

Most of the *behavioural aspects* are mapped into the process-protocols. In fact, the identified processes are almost directly adopted into process-protocols. The transitions of the EHORNET workflows are turned into AGACS, which also include the rules for their execution. In general, each process-protocol will either be executed by the entity which was created from the responsible actor (e.g. the child entity for the child workflow) or by an additional entity if the workflow cannot be clearly assigned to one of the identified actors. The latter might be the case in a more collaborative scenario, in which a workflow is executed by many different resources and not owned by one in particular. One important change from the EHORNET workflows to the process-protocol is the handling of communication. In the EHORNETS the communication is handled through merging the nets through named places. This is replaced in the PAFFIN-system with agent-like message communication. At each merging point an AGAC modelling the communication is added. This particular AGAC sends a message to the designated partner or receives and processes a message from a partner. This does not affect the structure of the process. For this paper's example the resulting process-protocols are *child* for the child entity and *mother/father* for the mother/father entities respectively. The places **mum/mum'/dad/dad'** are replaced with AGENT ACTIVITIES for communication.

To illustrate the mapping Figure 8 shows the original EHORNET and resulting father process-protocol. The recurring net structure in that figure is the $AGAC^9$. The inscriptions connect to the technical backend provided by the executing PAFFIN. At this point the coarseness of the mapping becomes evident. The mapping prescribes that merged places and transitions of the EHORNET system are mapped onto AGACS. However, as is visible in the example these AGACS are simple and contain only one agent action each (in sequence: receive message (from kid), think, create answer, send answer (to kid)). These actions could be better summarised into one AGAC, since they are semantically related. For now, issues like this are at the discretion of the modeller who has to decide which AGACS to combine and which to possibly extend. Capturing these reasonable combinations is part of the future work regarding the refinement of the mapping approach.

The *additional aspects* are mostly mapped into the knowledge and decision making mechanisms of the individual entities. The ontology is made known to the knowledge base of the PAFFINS. Adaption rules can be incorporated in two ways, depending on the extent of the changes. If the changes only affect one task/action, they are incorporated directly into the corresponding AGAC. AGACS support templates for actions, which are chosen depending on parameters defined in the inscriptions of the AGAC. If the changes are more extensive the adaption is moved to a decision making mechanism within the entity. This constantly active mechanism monitors the on-going process-protocols and can intervene and replace them if it decides upon this necessity. The process-protocols state is saved and a new process-protocol instantiated and moved to the previously saved state. In the example the latter is the case. The decision making mechanism can monitor the child process-protocol and, when the mother doesn't allow the candy, replace the process-protocol with the adapted version.

5 Related Work

The PAFFIN-system is directly related to [20, 27]. These works realised a **partial** integration of agents into a workflow management system. This partial integration allowed for workflows to exhibit agent properties (e.g. mobility, intelligence),

⁹ Please note that the irregular layout of the AGACS in relation to each other is intentional as to mirror the original HORNET.

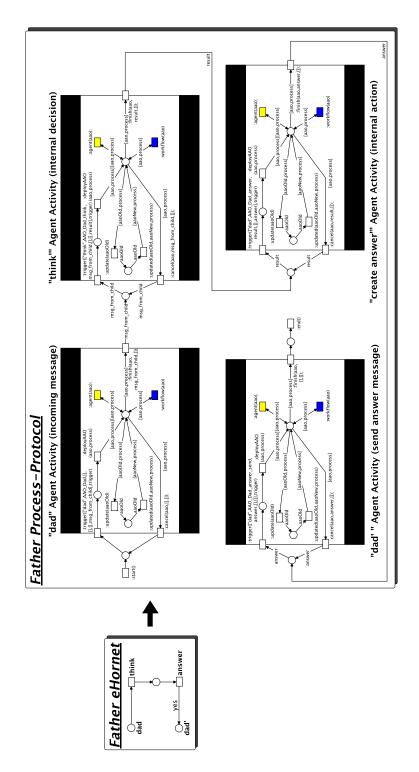


Fig. 8. The Father $\operatorname{EHORNET}$ and the corresponding Father process-protocol.

though modelling was restricted to workflows. The resulting outlook was the idea of a **full** integration, which the PAFFIN-system has now adopted.

Improvements of agents with workflow principles and workflows with agent principles, which we consider to be partial integrations, have been thoroughly researched before. For example, [2] uses agent intelligence to create an awareness of current workflows for an agent. By reasoning about most likely execution paths the agent can support users by providing better support. Other improvements of workflow management through agent properties can be found, e.g., in [19, 5, 23, 30]. A good overview can be found in [7], which identifies twenty-four functions of agents in workflow management and classifies over 100 publications into this scheme. Improving agents through workflow principles has been researched less often. Examples include [14, 15, 18]. Such efforts usually concentrate on designing and managing agent behaviour as workflows.

Such partial integrations do not offer the degree of integration the PAFFINsystem adopts. They still exclusively offer only agents **or** workflows as modelling constructs, and do not combine the concepts on the modeller's abstraction level. They do, however, use the respective other concept in the background, which allows some properties to be translated.

Adaptivity and flexibility in workflows have been the topic in many publications, e.g. [6]. Though the functionality is related to the topics of this paper, these works are mostly focused on the purely process and workflow aspects. Agents and an integration between them and workflows are not addressed.

There is some research that shows a larger degree of integration e.g. [9, 10, 31]. These approaches rely heavily on agents and their capabilities but still focus on processes as the main modelling abstraction. The PAFFIN-system does not stop there, but, as stated before, aims to provide a **full** integration.

6 Outlook and Conclusion

The paper presented a mapping between the EHORNETs formalism and the PAF-FIN-system, a hybrid agent and workflow system. The mapping retains aspects from the EHORNETs, such as mobility and adaptivity, while applying a formal basis to the hybrid PAFFIN-system.

The mapping creates a bridge between the two modelling approaches from which both can benefit. The EHORNETS are provided with a more practically oriented execution potential. PAFFIN applications are aimed at distributed systems with complex structures and behaviours. The possibility to make EHORNETS available in these kinds of systems is desirable. PAFFINS on the other hand benefit from the formal basis introduced by the EHORNETS. The ability to model an EHORNET system with certain formal properties (e.g. liveness, deadlock-freeness) and mapping that EHORNET system onto a PAFFIN application can make the use of PAFFINS more viable in practical situations. This paper presented the first step towards realising these advantages and benefits. The coarse mapping presented in this paper creates the foundation on which the practical mechanisms of PAFFINS and formal basis of EHORNETS can be incorporated.

Future work will address and build upon this coarse approach and refine it to fully exploit these intended benefits and advantages. Direct, short-term goals still focus on extending the modelling support and usability of the PAFFINsystem prototype. The long-term goals relate to actually exploiting the EHOR-NET mapping for purposes of verification. The ability to ascertain properties in the formal model (e.g. liveness, deadlock-freeness) for the practical execution is a highly desirable feature. Examining the conditions under which these properties are retained in the mapping and adapting the mapping to directly enforce these conditions are important results in the near future. Coupled with an envisioned, fully automated mapping between EHORNETS and the hybrid system, the results could be used to easily create formally verified practical systems. This future work also includes the vision of a reverse mapping from PAFFINS to EHORNETS. Such a mapping would make exploiting the formal basis for validation and verification introduced in this paper even simpler. Finally the extension from EHORNETS to HORNETS will allow more expressive models, but has to be balanced with the formal verification restrictions.

In conclusion, the presented mapping opens up the dynamic and formal nature of EHORNETs for a practical system. The PAFFIN-system itself combines agents and workflows on a fundamental and essential level. The idea to provide agents and workflows on the same level of abstraction naturally supports the complex relation between processes and the beings/objects that execute those processes in the real world. Consequently, the full integration proposed within the approach is a powerful, yet complex and difficult to handle mechanism. Establishing a formal basis for it is an important milestone towards the PAFFIN integration approach becoming a validated and well grasped modelling technique.

References

- van der Aalst, W.: Verification of workflow nets. LNCS 1248/1997, 407–426 (1997), application and Theory of Petri Nets 1997
- Both, F., Hoogendoorn, M., Mee, A., Treur, J., Vos, M.: An intelligent agent model with awareness of workflow progress. Applied Intelligence 36, 498–510 (2012), http://dx.doi.org/10.1007/s10489-010-0273-9
- Cabac, L., Dörges, T., Rölke, H.: A monitoring toolset for Petri net-based agentoriented software engineering. In: Valk, R., van Hee, K.M. (eds.) 29th International Conference on Application and Theory of Petri Nets, Xi'an, China. LNCS, vol. 5062, pp. 399–408. Springer-Verlag (2008)
- Cabac, L., Haustermann, M., Mosteller, D.: Renew 2.5 towards a comprehensive integrated development environment for Petri net-based applications. In: Kordon, F., Moldt, D. (eds.) 37th International Conference on Application and Theory of Petri Nets, Toruń, Poland. LNCS, vol. 9698. Springer-Verlag (Jun 2016)
- Czarnul, P., Matuszek, M., Wójcik, M., Zalewski, K.: BeesyBees efficient and reliable execution of service-based workflow applications for BeesyCluster using distributed agents. In: Proceedings of IMCSIT 2010. pp. 173 –180 (oct 2010)
- Dadam, P., Reichert, M., Rinderle-Ma, S., Göser, K., Kreher, U., Jurisch, M.: Von ADEPT zur AristaFlow BPM Suite - Eine Vision wird Realität: "Correctness

by Construction" und flexible, robuste Ausführung von Unternehmensprozessen. EMISA Forum 29(1), 9–28 (2009)

- 7. Delias, P., Doulamis, A., Matsatsinis, N.: What agents can do in workflow management systems. Artificial Intelligence Review 35(2), 155–189 (2011)
- Duvigneau, M., Moldt, D., Rölke, H.: Concurrent Architecture for a Multi-agent Platform. In: Giunchiglia, F., Odell, J., Weiß, G. (eds.) Agent-Oriented Software Engineering III. Third International Workshop, Agent-oriented Software Engineering (AOSE) 2002, Bologna, Italy, July 2002. Revised Papers and Invited Contributions. LNCS, vol. 2585, pp. 59–72. Springer-Verlag, Berlin Heidelberg New York (2003), http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2585&spage=59
- Guo, L., Robertson, D., Chen-Burger, Y.H.: A novel approach for enacting the distributed business workflows using BPEL4WS on the multi-agent platform. In: e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on. pp. 657 –664 (oct 2005)
- Hsieh, F.S.: Collaborative Workflow Management in Holonic Multi-Agent Systems. In: O'Shea, J., Nguyen, N., Crockett, K., Howlett, R., Jain, L.C. (eds.) Agent and Multi-Agent Systems: Technologies and Applications, LNCS, vol. 6682, pp. 383– 393. Springer-Verlag (2011), http://dx.doi.org/10.1007/978-3-642-22000-5 40
- Jacob, T.: Implementierung einer sicheren und rollenbasierten Workflowmanagement-Komponente f
 ür ein Petrinetzwerkzeug. Diploma thesis, University of Hamburg, Department of Computer Science, Vogt-K
 ölln Str. 30, D-22527 Hamburg (2002)
- Köhler-Bußmeier, M.: Hornets: Nets within nets combined with net algebra. In: Wolf, K., Franceschinis, G. (eds.) International Conference on Application and Theory of Petri Nets (ICATPN'2009). LNCS, vol. 5606, pp. 243–262. Springer-Verlag (2009)
- Köhler-Bußmeier, M., Heitmann, F.: Complexity results for elementary hornets. In: Colom, J.M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 150–169. Springer-Verlag (2013)
- Korhonen, J., Pajunen, L., Puustjärvi, J.: Using transactional workflow ontology in agent cooperation. In: AIM Workshop, First EurAsian Conference on Advances in ICT. Tehran, 2002 (2002)
- Kotb, Y.T.: Workflow-Net Based Cooperative Multi-Agent Systems. Ph.D. thesis, The University of Western Ontario, Electronic Thesis and Dissertation Repository (August 2011), paper 228
- Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002), http://www.logosverlag.de/cgi-local/buch?isbn=0035
- Kummer, O., Wienberg, F., Duvigneau, M., Cabac, L., Haustermann, M., Mosteller, D.: Renew – the Reference Net Workshop. Available at: http://www.renew.de/ (Jun 2016), http://www.renew.de/, release 2.5
- Mislevics, A., Grundspenkis, J.: Workflow based approach for designing and executing mobile agents. In: Digital Information Processing and Communications (ICDIPC), 2012 Second International Conference on. pp. 96–101 (july 2012)
- Pla, A., Gay, P., Meléndez, J., López, B.: Petri net based agents for coordinating resources in a workflow management system. In: ICAART 2011 - Proceedings of the 3rd International Conference on Agents and Artificial Intelligence, Rome, Italy, January 28-30, 2011. pp. 514–523 (2011)
- Reese, C.: Prozess-Infrastruktur f
 ür Agentenanwendungen, Agent Technology Theory and Applications, vol. 3. Logos Verlag, Berlin (2010)

- Reisig, W.: Petri nets and algebraic specifications. Theoretical Computer Science 80, 1–34 (1991)
- Rölke, H.: Modellierung von Agenten und Multiagentensystemen Grundlagen und Anwendungen, Agent Technology – Theory and Applications, vol. 2. Logos Verlag, Berlin (2004), http://logos-verlag.de/cgi-local/buch?isbn=0768
- Savarimuthu, B., Purvis, M., Purvis, M.: Different Perspectives on Modeling Workflows in an Agent Based Workflow Management System. In: Khosla, R., Howlett, R., Jain, L. (eds.) Knowledge-Based Intelligent Information and Engineering Systems, LNCS, vol. 3684, pp. 906–906. Springer-Verlag (2005), http://dx.doi.org/10.1007/11554028 30
- 24. Valk, R.: Petri Nets as Token Objects An Introduction to Elementary Object Nets. In: Desel, J., Silva, M. (eds.) 19th International Conference on Application and Theory of Petri nets, Lisbon, Portugal. pp. 1–25. No. 1420 in LNCS, Springer-Verlag, Berlin Heidelberg New York (1998), http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=1420&spage=1
- Valk, R.: Object Petri nets: Using the nets-within-nets paradigm. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Advanced Course on Petri Nets 2003. LNCS, vol. 3098, pp. 819–848. Springer-Verlag (2003)
- Wagner, T., Moldt, D.: Approaching the Integration of Agents and Workflows. In: Bergenthum, R., Desel, J. (eds.) 18. Workshop AWPN 2011, Hagen, September 2011. Tagungsband. pp. 55–61 (2011)
- Wagner, T., Quenum, J., Moldt, D., Reese, C.: Providing an Agent Flavored Integration for Workflow Management. In: Jensen, K., Donatelli, S., Kleijn, J. (eds.) Transactions on Petri Nets and Other Models of Concurrency V, LNCS, vol. 6900, pp. 243–264. Springer-Verlag, Berlin Heidelberg New York (2012)
- Wagner, T., Moldt, D.: Integrating agent actions and workflow operations. In: Müller, J.P., Ketter, W., Kaminka, G., Wagner, G., Bulling, N. (eds.) Multiagent System Technologies - 13th German Conference, MATES 2015, Cottbus, Germany, September 28-30, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9433, pp. 61–78. Springer (2015)
- Wagner, T., Moldt, D.: Workflow management principles for interactions between Petri net-based agents. In: Devillers, R., Valmari, A. (eds.) Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings. LNCS, vol. 9115, pp. 329– 349. Springer-Verlag (2015), http://dx.doi.org/10.1007/978-3-319-19488-2 17
- Y., L., C., L.: A workflow engine model based on multi-agent. In: Computer Application and System Modeling (ICCASM), 2010 International Conference on. vol. 14, pp. V14–486 –V14–489 (oct 2010)
- Zhaohui, L., Jia, C., Rui, G., Bin, X.: A Reconfigurable Platform of Manufacturing Execution System Based on Workflow and Agent. In: Software Engineering, 2009. WCSE '09. WRI World Congress on. vol. 1, pp. 314 –318 (may 2009)