Improving a Workflow Management System with an Agent Flavour

Daniel Moldt, José Quenum, Christine Reese, and Thomas Wagner

University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences,
Department of Informatics
http://www.informatik.uni-hamburg.de/TGI/

Abstract. This paper discusses an application of software agents to improve workflow management systems, with a practical emphasis on Petri net-based systems. The properties of agent technology will be used to gain advantages within the workflow management systems on both a conceptual and practical level. In this paper we discuss the theoretical background of our work, the conceptual idea and approach and one possible practical implementation. As a central practical means we use reference nets, a high-level Petri net formalism. These nets are used to model both agents and workflows, which results in a clean and natural integration of both technologies.

Keywords: High-level Petri nets, workflow management systems, multi-agent systems, software architecture.

1 Introduction

Workflows and *Workflow management systems* (WFMS) have been very attractive research topics in the last decades [13, 18, 16]. They provide means to further understand, unambiguously specify and analyse business processes within organisations. According to the state of the art, several heterogeneous WFMS can be combined to perform a specific complex task that cuts across various organisations. While such a combination is currently possible in an ad hoc manner, a more systematic approach demands greater care both from the modelling and implementation perspectives. In this research, we set out to address this limitation. This paper discusses the conceptual approach to achieve the overall goal.

Among the rising software paradigms, the concept of *agent* is a preeminent one. In the last decade, agents have been touted as a most appropriate paradigm to support the design and implementation of decentralised and distributed applications/systems, which yield intelligent behaviour and require a great deal of interoperability. A decentralised application implies an application which consists of autonomous entities. Clearly, these are among the properties one seeks while developing an approach for interorganizational workflows. Therefore, agents can

Recent Advances in Petri Nets and Concurrency, S. Donatelli, J. Kleijn, R.J. Machado, J.M. Fernandes (eds.), CEUR Workshop Proceedings, ISSN 1613-0073, Jan/2012, pp. 301–316.

contribute a great deal in our quest for a systematic approach for interorganizational workflows.

Using agents to design and implement distributed applications across a network is not new in itself. However, the autonomy of the various subparts is not well captured in the overall collaboration. As such, approaches of this kind cannot be generalised for the design and implementation of collaborative applications across various organisations. Concepts such as workflows, which render a clear view of business processes within organisations need to come into play.

Introducing agents in WFMS is not counterintuitive. By virtue of being autonomous, sociable and intelligent, human agents and artificial ones share many similarities. Moreover, human agents' operational mode can be viewed as a set of independent yet interoperable entities. From that angle, a human agent can be viewed as an agent system. Finally, since human agents have to coordinate the various tasks they are involved in at a certain point in time, they can be regarded as a WFMS. We take this human analogy, especially its duality, to show that both, agents and workflows, can be joined in a complex system.

As discussed in the foregoing, WFMS can benefit from agents in various regards. In this paper, we discuss seven points where agents help enhance the level of management in interorganizational workflows. These include distribution, autonomy, interoperability and intelligence. In order to make the resulting approach appealing to new technologies, we structure our assumptions and ideas into a reference architecture, which we believe lays the foundation for more specific and advanced architectures to support collaboration within and between organisations. We do not claim that our current implementation is as powerful as the existing commercial tools. However, thanks to the Petri nets formalism, our implementation holds the potential to properly handle concurrency, robustness and resilience in the future.

The contributions discussed in this paper are a clearer articulation of ideas and intuitions presented in [20–22]. More than all these papers, we elaborate on the underpinnings of the conceptual role agents can play in the new approach. Finally, we discuss the implementations.

The remainder of the paper comes as follows. Section 2 describes the technical and theoretical background of our work. Section 3 gives an overview of our overall architecture. Section 4 examines the conceptual view of our approach, while Section 5 discusses one implementation of our approach. Finally, Section 6 draws conclusions and directions for the future.

2 Frameworks & Formalisms

In this research, MASs are designed following Mulan's (**MUL**ti-**A**gent **N**ets) structure [11, 23]. Mulan has been extended with Capa (**C**oncurrent **A**gent **P**latform **A**rchitecture) in order to comply with Fipa's (**F**oundation for **I**ntelligent and **P**hysical **A**gent (see http://fipa.org)) (communication) standards to support concurrent execution [4]. Mulan and Capa describe the various components of a MAS using reference nets, which can be executed using the Re-

RENEW (**RE**ference **NE**ts **W**orkshop (see http://www.renew.de)) tool. The reader should thus note that not only do we offer a formal ground to reason about the behaviours of agents, but we also provide an execution environment for MAS.

Inspired from the *nets within nets* concept introduced by Valk [25], MULAN's structure is a four-layer architecture used to describe a MAS. These layers respectively describe the overall MAS, the agent platforms, the agents and their behaviour, the protocols. Every layer within MULAN's reference architecture is modelled using reference nets, a high level Petri net formalism which will be discussed later.

In order to adhere to FIPA's standards, and especially the communication mechanisms, MULAN has been extended with CAPA. In so doing, CAPA agents can easily interact with any type of FIPA compliant agents.

Each of the key concepts in this paper, agent and MAS on the one hand and workflows on the other hand, is represented using a Petri net formalism. Agents and MAS are represented using reference nets, while workflows and WFMSs are represented using workflow nets implemented as a special kind of reference nets. Reference nets have been introduced in [15]. They follow the philosophy of nets within nets. Reference nets use *Java* as an inscription language, manipulate various types of data structures, and like many other types of high-level Petri net formalisms, offer several types of arcs. Finally they use synchronous channels to synchronise with other nets or Java objects. The treatment of Java objects and net instances is transparent, so that both kinds of artefacts can be exchanged arbitrarily. The workflow nets used in our systems are based on principles of workflow nets introduced in [26]. They are implemented as reference nets and make use of a special task transition introduced in [9]. Moreover, in the Petri net community, tool support is a general trend. Therefore, the RENEW tool has been developed to support quick prototyping of systems or parts of systems using the reference net formalism. RENEW provides an editor to specify and draw the nets as well as a simulation engine to test and validate them.

3 Overall architecture

The results we present in this paper are part of a larger ongoing effort. The systems described in the next sections can be classified within the overall architecture described in [19]. The goal of this architecture is to integrate agent and workflow technologies. The architecture consists of five tiers, built on top of each other. Each tier itself is a layered architecture, which combines various aspects of both technologies. Starting from either a pure workflow or agent system, the architecture gradually evolves into a novel integrated unit system, which equally benefits from both original concepts. The main motivation in building this architecture lies in the shortcomings of each individual concept as is discussed in detail in [19] and [28]. In short, agent technology struggles with offering a clear behavioural view of large distributed systems, but can describe the structure of such a system in a natural way. Workflow technology can easily describe the behavioural view of a complex system, but struggles with the structural view.

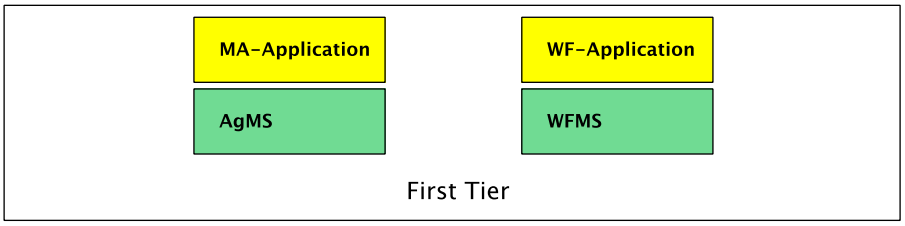

Fig. 1. Architecture of the first tier of the overall architecture, modified from [19]

In combining both technologies we can integrate the views offered by both into one new system which supports both a clean structural and behavioural view. It should be explained that the notion of tiers in this architecture denotes a kind of step-by-step refinement/enhancement on the way to the overall goal of integration. The tiers can be viewed as the layers of the overall *abstract* architecture but they do not correspond to layers within a *concrete* architecture. Each tier modifies the structure of its own layered architecture compared to the previous tier and in doing so enables new or improved aspects to be used. We will now shortly discuss the five tiers of the architecture.

First Tier The first tier is our starting solution to address the limitations of both technologies. It involves either a pure agent management system (AgMS) or a WFMS. Such systems exclusively use workflow or agent technology to provide their functionality. This means that there is almost no integration between the two. Because of this, the architectural view of this tier (see Figure 1) offers only two layers. The bottom depicts the adopted management system (either agent or workflow), on top of which an application lies. Examples of systems, which can be classified into this tier, are Mulan and Capa on the agent side and WFMS like ADEPT (see [3]) or WIFAi (see [24]) on the workflow side.

Second Tier In the second tier, one of the paradigms is used to realise the other. In other words, we use agents to design a WFMS, and vice versa. Because of this, there are two variants of the second tier (agents in the background or workflows in the background). The architectural view of this tier (see Figure 2) offers three layers. The topmost layer still represents an application but between the bottom management system and the application an intermediate layer has been added. This layer implements a management system for the alternative concept using the functionality of the bottom layer. For example, if the bottom layer is an AgMS, then the intermediate layer is a WFMS based on agents. The application layer of this tier uses *only* the concept provided by the intermediate layer.

Building on the first tier the second tier can be achieved by designing the application within the first tier to be the required management system. On top of that management system another application can then be built, turning the application layer of the first tier into the intermediate layer of the second tier.

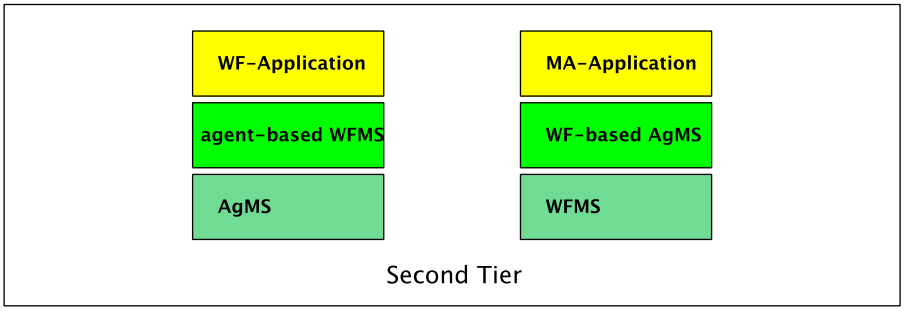

Fig. 2. Architecture of the second tier of the overall architecture, modified from [19]

Compared to the first tier, the advantage is that one can perceive an integration of the concepts. However, the available constructs only affect the background instead of being directly available in the application layer. For example, distribution, interoperability, etc. are facilitated in WFMS using agents.

In the remainder of this paper, the higher level tiers will only be based on the variation using agents in the background, i.e. the one including an AgMS, an agent-based WMFS (AgWFMS) and an application. Examples for these kind of systems are detailed in [6], [7] and [10].

Third Tier The third tier greatly enhances the application development by employing both agents and workflows. This results in an arbitrary degree of integration between agents and workflows. In addition to the interface between the application and intermediate layer, this tier allows direct access from the application layer to the bottom management system. In practice, the application can thus use both the interfaces offered by the core AgMS and the AgWFMS. Consequently, the key functionality of the AgMS is then combined with that of the AgWFMS. The architectural view of this tier (see Figure 3) only adds a direct connection between the application and the bottom layer, compared to the second tier. While this additional connection is a clear advantage over the previous tier by expanding potential and flexibility, it suffers from a major limitation. The resulting system is completely unstructured, i.e., the relation and integration between agent and workflow needs to be potentially re-invented for each application. Consequently it becomes very difficult to harness the power of this tier, especially the efficient design of complex systems. In order to reach a structured integration of both concepts within the architecture, we need to take one step back and limit the immense possibilities offered by this tier.

Fourth Tier The fourth tier adds an integration layer to the architecture, which is responsible for restricting the possibilities of the third tier in order to provide an explicit structure for the application developed on it (see Figure 4). Through this integration both technologies are used in the background and the relationship between agents and workflows is pre-defined. However, only *one* perspective is

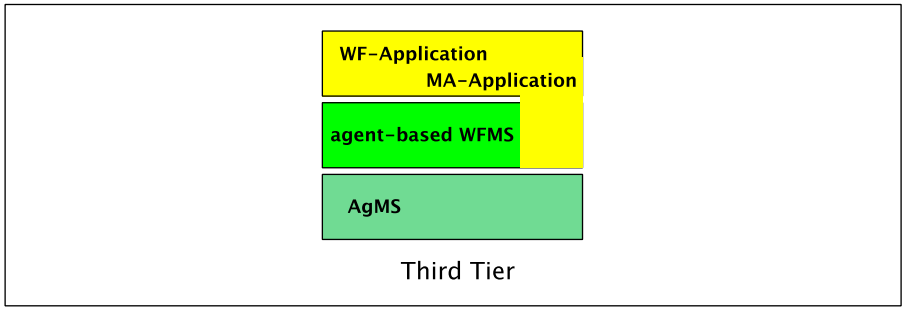

Fig. 3. Architecture of the third tier of the overall architecture, modified from [19]

supported when modelling on the application layer. In this way it provides an abstraction and thus a higher level of modelling.

In order to achieve the desired explicit structure, this tier basically reduces the functionality offered to the application layer compared to the third tier. It refocuses on either agents or workflows as the exclusive main abstraction for application development. There are once again two distinct variations of this tier, one offering agents to the application (called *workflowagents*; left hand side of Figure 4), the other one offering workflows (called *agentworkflows*; right hand side of Figure 4). The integration layer provides exclusively WFMS or AgMS functionality, but uses both technologies in the background. This means that an agent application possesses parts and aspects of workflows and vice versa (in the other variation). In this way the possibilities of this tier are, opposed to the third tier, restricted, because we refocus on just one technology. But by doing this, we gain a much more powerful means of supporting one of the two technologies. The integration of both variations will take place in the fifth tier. For now we need both variations in order to create the desired structure within the relation between agents and workflows in both directions separately.

It is worth noting that, even though we are looking at either workflows or agents at the top layer again, the main difference between this tier and the second tier is that we no longer have one concept realising the other. Rather, we obtain a successful combination of both, i.e., agents and workflows working side by side and benefiting from each other. The agentworkflow variation of this tier is the main focus of this paper and will be discussed in detail in the main sections.

Fifth Tier This tier introduces the concept of *unit*, an abstraction to any entity involved in the design of the system. Units offer both the facets of agents and workflows. In order to achieve this, the AgMS and WFMS have to be integrated and combined. This results in a novel type of management system, a *unit management system* (UMS). The architecture of this tier can be seen in Figure 5. The integration layer of the fourth tier has been split into two parts, of which UMS represents the upper part. Since one can no longer clearly differentiate between

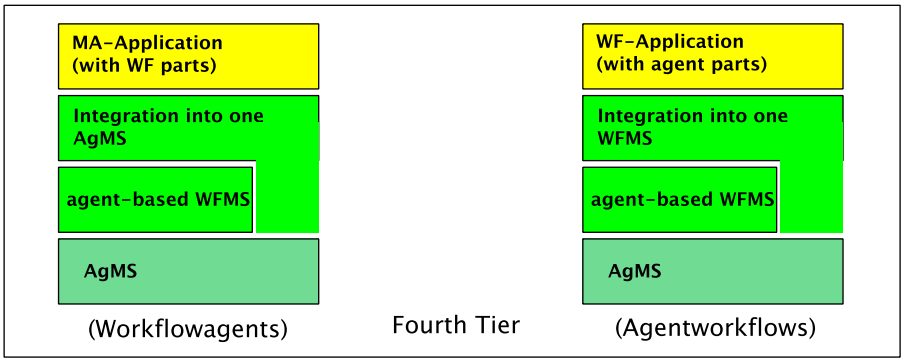

Fig. 4. Architecture of the fourth tier of the overall architecture, modified from [19]

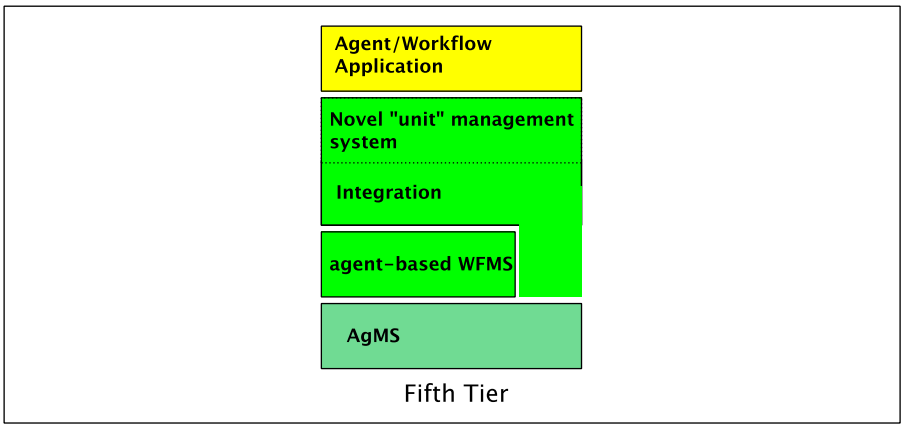

Fig. 5. Architecture of the fifth tier of the overall architecture, modified from [19]

agents and workflows, the application layer is simply called unit application, also referred to as agent/workflow applications in [19]. In merging both agent and workflow concepts into a single unit concept, both the structural view (from the MAS) and the behavioural one (from the workflows) are available. Note that both these views are available during runtime and design time.

4 Conceptual View

In this section we will discuss our conceptual approach to improving workflow management with agent technology. As stated before our goal is to use aspects of software agents to benefit the execution and management of workflow instances. Because we use agents to improve workflows we generally refer to this approach as *agentworkflows*. We reason that common properties of agents, like mobility,

autonomy and proactivity, and especially the encapsulation of workflow instances through agents can greatly benefit workflow execution. Having agents in general handle aspects of workflow management can also help distribute the execution in order to spare less powerful resources.

As mentioned before this particular work is just part of a larger effort to integrate workflow and agent technologies in order to achieve a novel approach that combines the advantages of both technologies. The work presented in this paper focuses on the fourth tier of the overall architecture (see Section 3). We discuss the variation of the fourth tier, in which the integration layer offers a WFMS to the application layer. This WFMS strongly relies on the functionality of the AgMS in the background in order to provide its own functionality. This means that the workflows offered to the application possess some properties gained from the agents. How this can be achieved will be discussed in this section, as well as the advantages and disadvantages of this approach,.

One of the core ideas behind our approach is to encapsulate workflow instances through autonomous software agents. With this it is possible to transfer properties of that software agent directly to the workflow instance. In other words the clear separation between agent and workflow begins to diminish. This is a key concept of the overall architecture and is important for the fifth tier. Another important aspect of our approach is to also consider other entities of the workflow management system as agents. Having the general functionality of the WFMS be provided by agents is already part of an AgWFMS of the second tier. In our approach agents can be used to realise any of the elements (e.g., tasks, users, resources, etc.) of the workflow as well. In [17] for example, we used agents to realise activities. Thanks to the agent technology, the resulting WFMS does not only focus on the behaviour of the system, as was the case in the previous tiers of the overall architecture, it also emphasises the structure of the system. This paper focusses on the aspect of encapsulating workflow instances through agents. Other aspects are considered but will not be discussed in great detail.

We will now examine some of the principal and conceptual areas and aspects in which workflows can benefit from agents. The following points will directly cover some agent properties and their particular benefits but will also include some general observations.

Encapsulation In general the encapsulation of one or more workflow instances through agents can be seen as a prerequisite to opening up many of the possibilities offered by the agent-oriented paradigm. Without this concept it would be hard or impossible to transfer other agent properties over to the workflows. Nonetheless the encapsulation also benefits the workflows in more ways than that. For example the encapsulation provides the workflows with an even clearer identity within the overall system since they can now be identified in the same way as the other elements (agents) of the system. This makes it easier to monitor, observe and analyse the system, which in turn makes maintenance and improvement more efficient. A disadvantage of the encapsulation is that the number of agents active is possibly, drastically increased, depending on how the encapsulation is handled. This may pose

problems on less powerful systems, which simply cannot handle this number of agents or the communication between them. However, since agent architectures are generally built to efficiently handle communication, this should not pose a real problem in practical use.

Mobility By allowing workflows to gain agent mobility they benefit in a variety of possibilities. In the context of software agents mobility describes the capability of a software agent to discontinue its execution within one execution environment (agent platform), migrate to another environment and continue the execution there starting off from its previous state. For agentworkflows this means that the execution of a workflow can be discontinued on one instance of an executing WFMS and continued on another WFMS. Practically this can be used if certain resources needed for the execution of a workflow are not available on every platform. This can include particular (groups of) users or certain, possibly critical data. Another use case for this property is to have a workflow instance migrate not because certain resources are needed, but because its home platform is beginning shutdown or because another platform carries less of a load then the home platform. This use of mobility can lead to improved flexibility, efficiency and fault tolerance.

Autonomy One of the key concepts of the agent paradigm is that agents are autonomous entities. This means that, to a certain degree, they are independent of their environment and can choose for themselves whether to execute an action or not. In the context of agentworkflows this property can be used in a number of ways. It can for example be used as a kind of access control to critical data for which an agent is responsible. This can be a workflow instance but also other entities like activities or the handling of users. Another use of this becomes relevant if combined with mobility. An agent migrating to another platform to access certain data or perform certain actions can do this relatively independent from the other agents and software constructs of that platform, if, of course, it has all the necessary permissions.

Intelligence Intelligence in software agents can be used to describe a multitude of aspects. One major aspect is the ability of certain agents to proactively decide by themselves which actions to take. In the context of workflow management this can be used to predetermine which users should be offered certain tasks, taking variables like workload into consideration. At this point reactivity of agents also comes into play. Software agents can react to events in their environment and adapt according to the situation. For example if there is an error during the execution of a task the agent could observe this and retry the action with changed parameters. Another very interesting aspect where intelligence, proactivity, reactivity and adaptiveness can be used is the adaptivity of workflow instances. Changing workflow instances and even entire workflow definitions according to changed circumstances (current or permanent) improves the flexibility and versatility of a WFMS and can be handled in a natural way using agent intelligence.

Distribution The agent oriented paradigm naturally supports the design of distributed software systems. The main reasons for this are the asynchronous message communication and the autonomy of the individual agents. By re-

lying on agents as the main building blocks of a WFMS it is easy to use these predispositions for the distribution of the system. The communication of different parts of the system can be handled through asynchronous messages, which are flexible and versatile. Extending on this idea opens up even more possibilities of using distribution to the advantage of workflows. Interorganizational workflows can benefit from a distributed WFMS, so that their critical information is not stored in some centralised location.

Interoperability The Fipa communication standards are accepted by many widely-used agent frameworks. Adhering to these standards guarantees interoperability between the different involved software systems, independent of agent architecture or framework. This can be translated into workflow management based on agents as well. In this case different WFMS of different providers can work together, as long as they can process the data structures that are exchanged. This aspect is especially important in the context of interorganizational workflows since it allows some freedom for the choice of the different WFMS in the different companies. But also in general use cases interoperability can be used to an advantage. A Fipa-compliant WFMS can request data from any other Fipa-compliant system, which improves the possibilities of the WFMS. Another aspect which is related to this and distribution, is the openness of the system. Through interoperability and distribution it is possible to create flexible and dynamic open systems to which different WFMS can connect to, complete some tasks, and then disconnect again. Open systems can provide users with functionality that is otherwise difficult to obtain without specialised software solutions.

Structure This point is related to the motivation behind our overall architecture. As described, we reason that workflow systems have trouble adequately describing the structure of the system they are modelling, while focussing on the behaviour. Agent systems on the other hand possess a strong focus on this structure. By joining the two in the ways described in this paper we begin to combine this structural view given by the agents with the behavioural view of the workflows. This is mostly related to the encapsulation aspect discussed above, but contains a more abstract view. By relying on agents one can easily describe the current state of an entity including its current location (in regards to distribution), knowledge and behaviour. By adapting this for workflow instances it can already help provide the structural view needed within a distributed system. If the agentworkflow idea is taken even further and every aspect of the system modelled through agents the structural view becomes even more useful. The location (in regards to distribution) of every resource, user and workflow can be determined and displayed in a way that helps monitoring and maintaining the system.

It should be noted that all these properties only unfold their full potential if used in combination. Every one of these properties and aspects possesses some benefits but together with the others new and improved possibilities can be achieved. For example using mobile agents in a distributed environment of many interoperable agent platforms is more advantageous then forcing the same agent

system onto all involved partners. Equally an autonomous, intelligent agent can decide for itself if a migration is reasonable or not and initiate the action accordingly. Using these properties together also strongly improves interorganizational workflow management and execution. While interoperability and distribution already favour this field, the other properties are also useful, especially in collaboration. For example mobility allows for the transmission of data in a natural way, while encapsulation allows for the clear separation of critical data.

The main disadvantage of our approach is that the realisation and handling of these improved workflows are more complex than handling regular workflows. The reason for this is mainly that the new and improved possibilities will be difficult to harness. It can, if used in the wrong way, affect execution in a negative way or even, in the worst case, prohibit correct execution at all. However, if used correctly and efficiently, they offer clear, distinct advantages to workflow execution in general. They offer novel ways of modelling many parts of workflows and can increase efficiency in use.

After discussing the conceptual view in this section we have shown that our approach offers many advantages, but is difficult to realise and handle. For the realisation part we have chosen technologies based on Petri nets. One problem of the conceptual approach is that different kinds of entities (agents and workflows) have to be combined. By choosing Petri nets as a common basis we can partially circumvent this problem, since it is easier to combine the two kinds of entities when they possess the same basis at the lowest level. On the other hand this choice has the problem of not being widely spread and available. However, certain aspects, like concurrency and displaying behaviour, are very easy to model using Petri nets. In the next section we will discuss one prototypical implementation of our conceptual approach using Petri nets, which already covers some of the properties described in this section.

5 Implementation

In this section we will discuss a prototypical implementation of our agentworkflow approach. As mentioned before we use Petri net based technologies to achieve a common basis for the integration and combination of workflow and agent technologies. In particular we use MULAN and CAPA for our agents and workflow nets for our workflow functionality (see Section 2). The starting point of the practical work is an AgWFMS of the second tier of the overall architecture. This AgWFMS has been described in detail in [27]. It relies solely on agents to provide the functionality but does not mix the agent and workflow concepts enough to be considered an agentworkflow system.

Before going into the details of the implementation we will shortly discuss how the different properties observed in the conceptual section can be mapped onto Petri nets. Since discussing these aspects in a reasonable extent would go beyond the scope of this paper and since extensive work on this has already been performed and published we will limit this to referring to other contributions. The mobility aspect has been extensively studied, especially in the context of

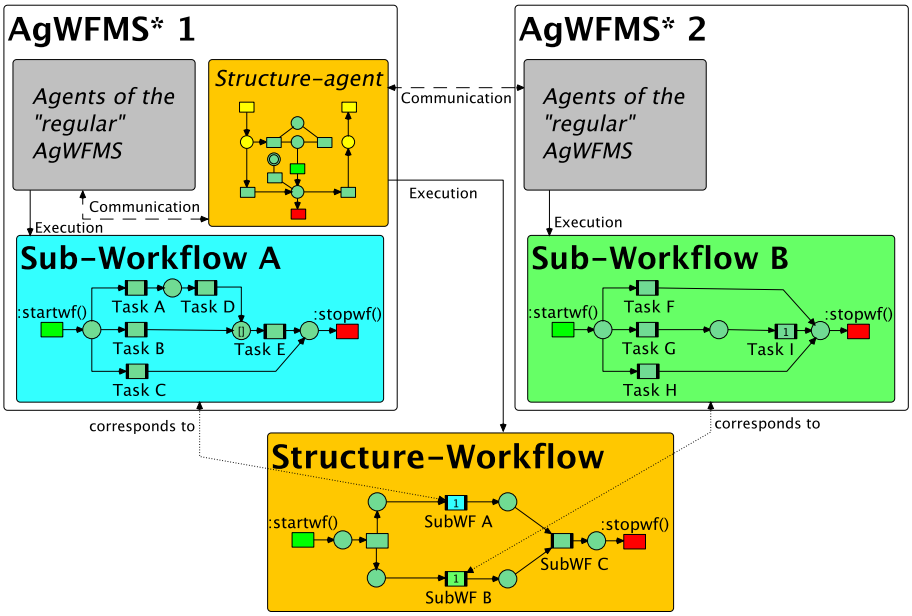

Fig. 6. Principle approach of the S-AgWf

nets within nets, for example in [2] and [12]. The autonomy and intelligence of Petri net agents have been discussed in the context of Mulan in [23]. The encapsulation aspect has been examined for object-oriented nets in [1]. Interoperability and openness have been explored in [14]. The final aspect, the structural view of combining agents and workflows, was discussed in [19].

The practical approach, called structure-agentworkflow (S-AgWf), extends the regular AgWFMS (now called AgWFMS*) to allow for the definition and execution of distributed workflow instances. More precisely, the workflow instances are now hierarchical workflows with nested subprocesses as defined by the WfMC (see [8]).As a consequence the entire system consists of a number of Capa platforms, which all execute instances of the extended AgWFMS* and are working together. The different AgWFMS* instances are known to one another and messages can be exchanged between them. A more detailed description of the S-AgWf approach can be found in [28].

The basic principle of the S-AgWf can be seen in Figure 6. One agent encapsulates one workflow instance. This agent, called structure-agent, possesses an internal workflow, the structure-workflow. When the structure-agent for a new workflow instance is started, it receives the definition of the structure-workflow from the database agents of the AgWFMS* and instantiates the workflow net. When this initialisation is finished, the execution of the structure-workflow automatically begins. The tasks of the structure-workflow correspond to sub-workflows. Sub-workflows can only be executed on certain AgWFMS* instances

within the overall system. The information about which AgWFMS* instance is suitable is stored within the data of the task and can be extracted by the structure-agent. Whenever a task becomes active, the structure-agent assigns itself as the executor of that task. Once this is done the structure-agent queries a special agent of his own platform for a list of all the known AgWFMS* instances currently active. It then compares this list to the information extracted from the task and chooses a suitable AgWFMS* instance. The interface-agent of the chosen instance is then contacted by the structure-agent. The structure-agent asks the interface-agent to instantiate the subworkflow locally and transmits all relevant parameters, like input data etc. The subworkflow is then executed like any other workflow in the regular AgWFMS. Once it has reached the end of its execution the responsible structure-agent is informed and any (optional) results are sent back. The results are transmitted into the structure-workflow net and the structure-agent completes the task, so that the execution can continue. The execution of tasks and subsequent instantiation and execution of sub-workflows is continued, until the end of the structure-workflow is reached. The initiator of the overall workflow is then informed and the structure-agent can terminate.

In the example in Figure 6 two sub-workflows are currently executed on the two different AgWFMS* platforms. The structure-agent responsible for the structure-workflow is communicating with the agents of the two AgWFMS* platforms in order to initiate the execution and receive results. When both sub-workflows are finished the structure-agent will start a final sub-workflow ($SubWF\ C$), before it can conclude the execution of the structure-workflow.

This realisation of the agentworkflow concept offers distinct and practical advantages, but also still suffers from some limitations. The possibility to distribute the execution of workflows is a huge advantage for the otherwise centralised AgWFMS. The support of nested subprocesses allows for interorganizational workflows to be defined and executed. Since the details of the local workflows are not needed globally, the sub-workflows and any critical data they may contain are only known to the local parties. This satisfies the need of interorganizational workflows to secure and conceal confidential and valuable information.

The main limitation of this particular, specialised implementation of the agentworkflow concept is its still centralised nature. If the platform of the structure-agent is disconnected or fails, the entire workflow fails. This could partially be rectified by adding mobility to the structure-agent. It can then easily migrate to another platform, if it discovers any changes in its home platform that might hinder its execution.

The pre-defined relationship between agents and workflows within this system combines the structural aspect of agents with the behavioural aspect of workflows as is the goal in the fourth tier of the overall architecture. The two concepts agent and workflow begin to merge together, since in this system a workflow is an agent and partly vice versa. The practical advantages this particular system gains from this merge mostly consist in a groundwork for further enhancements. Agent autonomy may for example be used to give the structure-agent more control over the workflow instance (e.g. choice over where

sub-workflows are executed), which could result in added flexibility. However the instances within the S-AgWF system already possess certain degrees of distribution (structure-agents communicate with other AgWFMS* platforms in order to execute their structure-workflow), interoperability (the structure-agents exchange FIPA-compliant messages so it is possible to exchange the AgWFMS* platforms with other WFMS if they adhere to the interface) and encapsulation (the structure-workflow is clearly encapsulated by the structure-agent).

6 Conclusion

In this paper, we made a strong case for a systematic introduction of the agent concept to enhance the management of workflows. We pointed out key aspects in which agents improve workflows and their management.

In our quest to develop a systematic approach to support workflow management, we proposed a reference architecture, which builds on an integration of agents and WFMS. The architecture consists of five distinct tiers. These different tiers gradually show how the agent concept first integrates into WFMS and then enhances them on the various aspects we discussed above. This effort culminates in the fifth tier, where both concepts exist alongside each other. As stated in the foregoing, our overall goal in this research is to achieve a seamless integration of agents and WFMS. In this paper, we presented an approach which builds on the agent technology to address WFMS. However, the other variant of the fourth tier, the workflowagents, needs to be considered as well. In it, the main abstraction of the application layer are agents, which strongly rely on the functionality of the WFMS in the background to address the inherent limitations to an agent-based system. Clearly, following that perspective, a WMFS could for example bring its systematic and proof-driven approach to complex task execution. In the future, we wish to explore that perspective as well.

From the lessons learned from both approaches, we expect to collect the amount of information that enables us to design a full-fledged conceptual approach which offers the best of both agent and workflow technologies in one single system, i.e. the fifth tier. Such an approach will balance out the weaknesses of each technology. With the support of high-level Petri nets as a foundational formalism, we are guaranteed of combining structure and behaviour in one representation.

References

1. Ulrich Becker and Daniel Moldt. Objekt-orientierte Konzepte für gefärbte Petrinetze. In Gert Scheschonk and Wolfgang Reisig, editors, *Petri-Netze im Einsatz für Entwurf und Entwicklung von Informationssystemen*, Informatik Aktuell, pages 140–151, Berlin Heidelberg New York, 1993. Gesellschaft für Informatik, Springer-Verlag.
2. Lawrence Cabac, Daniel Moldt, Matthias Wester-Ebbinghaus, and Eva Müller. Visual Representation of Mobile Agents – Modeling Mobility within the Prototype MAPA. In Duvigneau and Moldt [5], pages 7–28.

3. Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, and Martin Jurisch. 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.
4. Michael Duvigneau. Bereitstellung einer Agentenplattform für petrinetzbasierte Agenten. Diploma thesis, University of Hamburg, Department of Computer Science, Vogt-Kölln Str. 30, D-22527 Hamburg, December 2002.
5. Michael Duvigneau and Daniel Moldt, editors. *Proceedings of the Fifth International Workshop on Modeling of Objects, Components and Agents, MOCA'09, Hamburg*, number FBI-HH-B-290/09 in Bericht. University of Hamburg, September 2009.
6. Lars Ehrler, Martin Fleurke, Maryam Purvis, and Bastin Tony Roy Savarimuthu. Agent-based workflow management systems (WfMSs) - JBees: a distributed and adaptive WfMS with monitoring and controlling capabilities. *Information Systems and E-Business Management*, 4, Number 1 / January, 2006:5–23, 2005.
7. Andrea Freßmann, Rainer Maximini, and Thomas Sauer. Towards Collaborative Agent-Based Knowledge Support for Time-Critical and Business-Critical Processes. In *Professional Knowledge Management*, volume 3782, pages 420–430, Berlin Heidelberg New York, 2005. Springer-Verlag.
8. David Hollingsworth. *The Workflow Reference Model*. Workflow Management Coalition. Verfügbar auf http://www.wfmc.org/reference-model.html.
9. Thomas Jacob. Implementierung einer sicheren und rollenbasierten Workflowmanagement-Komponente für ein Petrinetzwerkzeug. Diploma thesis, University of Hamburg, Department of Computer Science, Vogt-Kölln Str. 30, D-22527 Hamburg, 2002.
10. N. R. Jennings, T.J. Norman, and P. Faratin. ADEPT: An Agent-Based Approach to Business Process Management. *ACM SIGMOD Record*, 27:32–39, 1998.
11. Michael Köhler, Daniel Moldt, and Heiko Rölke. Modelling the Structure and Behaviour of Petri Net Agents. In J.M. Colom and M. Koutny, editors, *Proceedings of the 22nd Conference on Application and Theory of Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 224–241. Springer-Verlag, 2001.
12. Michael Köhler, Daniel Moldt, and Heiko Rölke. Modelling mobility and mobile agents using nets within nets. In Wil van der Aalst and Eike Best, editors, *Proceedings of the 24th International Conference on Application and Theory of Petri Nets 2003 (ICATPN 2003)*, volume 2679 of *Lecture Notes in Computer Science*, pages 121–139. Springer-Verlag, 2003.
13. Michael Köhler-Bußmeier. Hornets: Nets within Nets combined with Net Algebra. In Karsten Wolf and Giuliana Franceschinis, editors, *International Conference on Application and Theory of Petri Nets (ICATPN'2009)*, volume 5606 of *Lecture Notes in Computer Science*, pages 243–262. Springer-Verlag, 2009.
14. Michael Köhler-Bußmeier. SONAR: Eine sozialtheoretisch fundierte Multiagentensystemarchitektur. In Rolf v. Lüde, Daniel Moldt, and Rüdiger Valk, editors, *Selbstorganisation und Governance in künstlichen und sozialen Systemen*, volume 5 of *Reihe: Wirtschaft – Arbeit – Technik*, chapter 8–12. Lit-Verlag, Münster - Hamburg - London, 2009.
15. Olaf Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.
16. Kolja Markwardt, Daniel Moldt, and Christine Reese. Support of Distributed Software Development by an Agent-based Process Infrastructure. In *MSVVEIS 2008*, 2008.

17. Kolja Markwardt, Daniel Moldt, and Thomas Wagner. Net Agents for Activity Handling in a WFMS. In Thomas Freytag and Andreas Eckleder, editors, *16th German Workshop on Algorithms and Tools for Petri Nets, AWPN 2009, Karlsruhe, Germany, Proceedings*, CEUR Workshop Proceedings, 2009.
18. Daniel Moldt. *Höhere Petrinetze als Grundlage für Systemspezifikationen*. Dissertation, University of Hamburg, Department of Computer Science, Vogt-Kölln Str. 30, D-22527 Hamburg, August 1996.
19. Christine Reese. *Prozess-Infrastruktur für Agentenanwendungen*. Dissertation, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg, 2009. Pdf: http://www.sub.uni-hamburg.de/opus/volltexte/2010/4497/.
20. Christine Reese, Jan Ortmann, Daniel Moldt, Sven Offermann, Kolja Lehmann, and Timo Carl. Architecture for Distributed Agent-Based Workflows. In Brian Henderson-Sellers and Michael Winikoff, editors, *Proceedings of the Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005), Utrecht, Niederlande, as part of AAMAS 2005 (Autonomous Agents and Multi Agent Systems), July 2005*, pages 42–49, 2005.
21. Christine Reese, Matthias Wester-Ebbinghaus, Till Dörges, Lawrence Cabac, and Daniel Moldt. A Process Infrastructure for Agent Systems. In Mehdi Dastani, Amal El Fallah, Joao Leite, and Paolo Torroni, editors, *MALLOW'007 Proceedings. Workshop LADS'007 Languages, Methodologies and Development Tools for Multi-Agent Systems (LADS)*, pages 97–111, 2007.
22. Christine Reese, Matthias Wester-Ebbinghaus, Till Dörges, Lawrence Cabac, and Daniel Moldt. Introducing a Process Infrastructure for Agent Systems. In Mehdi Dastani, Amal El Fallah, João Leite, and Paolo Torroni, editors, *LADS'007 Languages, Methodologies and Development Tools for Multi-Agent Systems*, volume 5118 of *Lecture Notes in Artificial Intelligence*, pages 225–242, 2008. Revised Selected and Invited Papers.
23. Heiko Rölke. *Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen*, volume 2 of *Agent Technology – Theory and Applications*. Logos Verlag, Berlin, 2004.
24. Michael Tarullo, Daniela Rosca, Jiacun Wang, and William Tepfenhart. WIFAi - A Tool Suite for the Modeling and Enactment of Inter-organizational Workflows. In *SOLI '09. IEEE/INFORMS International Conference on Service Operations, Logistics and Informatics 2009*, pages 764–769. IEEE, 2009.
25. Rüdiger Valk. Concurrency in Communicating Object Petri Nets. In *Advances in Petri Nets: Concurrent Object-Oriented Programming and Petri Nets*, volume 2001 of *Lecture Notes in Computer Science*, pages 164–195. Springer-Verlag, Berlin Heidelberg New York, 2001.
26. Wil M.P. van der Aalst. Verification of Workflow Nets. In *ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, volume 1248, pages 407–426, Berlin Heidelberg New York, 1997. Springer-Verlag.
27. Thomas Wagner. A Centralized Petri Net- and Agent-based Workflow Management System. In Duvigneau and Moldt [5], pages 29–44.
28. Thomas Wagner. Prototypische Realisierung einer Integration von Agenten und Workflows. Diploma thesis, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg, 2009.