Abstract

Within open distributed systems the realization of a spanning application is an open problem. While the local functionality can be implemented based on established approaches, the overall control of the processes to form a consistent and correct application remains difficult. Workflow management systems (WFMS) are one solution for process control. In combination with distributed systems further issues have to be solved, and are here tackled under different perspectives like Petri nets (to provide a true concurrency semantics of the concepts) and agents (to provide a powerful middleware and a more abstract modeling paradigm than objects or components).

In this paper we coin the phrase process infrastructure. The idea is to provide all means to model, build, control and maintain the processes within open agent networks as special distributed systems by combining the above mentioned concepts and technical means. To gain such a powerful process infrastructure, we started to build prototypes, which stepwise introduce some implementations of the advanced concepts. The potential of our proposed solution lies in the flexibility, rigor formally precise and hence direct executable approach, which is introduced by the autonomous and adaptive handling of processes in specific units (agents), which use and produce the necessary infrastructure to handle processes in different contexts at all levels.

1 Introduction

Collaborative business scenarios raise the question of how to integrate the cooperating enterprises. One has to deal with cross-organizational processes characterized by distributed entities that have distinct (and purposefully hidden) local knowledge in addition to the common global knowledge within the network. The corresponding localization of data, behavior, and decision making requires an exact conceptualization of the organization-spanning business processes that explicitly includes the participants’ information and communication technologies.

Several authors (see section Related Work) propose to integrate the concepts of agents, workflows and traditional computer science techniques to build better applications. Multi-agent applications represent an important subclass of such distributed, concurrent and large applications. The overall goal of our work is to improve the development of multi-agent systems. In [1][2][3] we presented the
conceptual framework. Now we present the concept of a process infrastructure for open agent networks together with an implemented workflow agent as its important component. The workflow agent encapsulates a process to allow for autonomous and flexible handling of system spanning processes, enabled through the process infrastructure. The process infrastructure facilitates the instantiation, maintenance, modification, and termination of processes at the conceptual and the technical level.

We start with an approach to processes, workflow and multi-agent systems (MAS) and the basic concepts in Sections 2 and 3. The specification of our process infrastructure is presented in Section 4, followed by a short glimpse of a running prototype and application examples in Section 5.

2 Processes, Workflows and Multi-Agent Systems

First of all, we present our terminology and outline the underlying concepts and their interconnections. Then we elaborate on related work in the areas of (distributed) workflow management systems and present concepts of the area of agent networks.

2.1 Terminology

In the area of system-spanning cross-organizational process management, our focus is on open agent networks. We provide a generic infrastructure for the support. This infrastructure is made up of agents as the technical term for flexible and autonomous encapsulated entities with problem-solving capabilities. Agents make up Multi Agent Systems which are our goal technology environment. The agents draw on well-established mechanisms of workflow management and enrich those.

Processes and Petri Nets We use Petri nets for both modeling and programming purposes. Petri nets have an operational semantics that makes them specifically tailored for process-oriented models. They are a well-known means for modeling the concepts of concurrency, independence, precedence and conflict when regarding activities. The term process has a specific meaning in Petri net terminology. Petri net processes are a recognized alternative for describing the behavior of Petri nets by firing sequences. Processes are themselves Petri nets from the class of causal nets, where no forward branching is allowed for the places. We refer to [4] for a thorough introduction to Petri nets.

Reference nets (introduced by Kummer in [5], in German) as a higher-order Petri net formalism show some extensions compared to conventional colored Petri nets. They implement the nets-within-nets paradigm where a surrounding net (the system net) can have nets as tokens (the object nets). To facilitate communication between nets, synchronous channels permit a fusion of two transitions at a time for the duration of one occurrence and thus enable bi-directional information flows. In addition, reference nets may carry complex JAVA inscriptions.
and consequently offer the possibility of Petri net-based Programming using the multi-formalism Petri net editor and simulator RENEW as an integrated development environment.

**Agents** The Petri net-based multi-agent framework MULAN (introduced by Rölke et al. in [6]) allows for the modeling and execution of multi-agent applications with reference nets. The Capa extension by Duvivneau et al. [7] expands the MULAN specifications to reach FIPA-compliance. Agents are hosted on multiple platforms that are connected through a technical communication infrastructure and together make up the multi-agent system as a whole. Agents communicate in terms of the agent communication language FIPA-Acl and domain specific ontology. One factor defining the interrelationships between the agents is services, where agents can be service providers or consumers. We deal with different types of agents according to the roles they embody and the services they offer. The agents use a (distributed) directory service in order to identify and contact companions.

**Workflow Management Systems** A workflow is “the computerized facilitation or automation of a business process, in whole or part” as defined by the Workflow Management Coalition (WfMC [8]) in the workflow reference model. A Workflow is composed of tasks, an executable task with case data is called a workitem and a workitem that is assigned all necessary resources and a user to do the work is called an activity. Especially interesting are management systems for distributed or inter-organizational workflows and a plenty of products, researches and projects exist that address this aspect (see Related Work).

One especially suited possibility to handle the arising complexity of such systems is agent-based workflow management. Another basic way to handle complexity is to use nested structures, as it is possible with the nets-within-nets concept provided by reference nets. Additionally, Petri nets are especially suited to represent workflows. Van der Aalst introduced Workflow Petri Nets and published on verification and analysis of those: [9][10][11]. Jacob used reference nets to design and implement a workflow management system [12], building on concepts elaborated by Moldt et al. in [13].

### 2.2 Related Work

The general aim to support distributed, heterogeneous, or open systems is addressed from several directions in different approaches. We believe that the outcome is converging to general flexible infrastructures providing support for processes. Grid technology combined with Service Oriented Architectures is one of those approaches. For example Böhme and Saar [14] address integration of services based on different communication frameworks, languages and platforms in order to develop an open software platform for adaptive services discovery, creation, composition, and enactment. They do not look at FIPA-agents although they are easy to integrate since agents can also provide services described in
WSDL. Their notion of workflow is not stressed at the management part as specified by the WfMC. Distributed workflows related to Grid technology is researched for example by Burchard et al. [15]. In this area, it is important to distinguish between concepts and realization of concepts. The concepts for agents and Web Services are different, the realization is crosswise possible. For example, Blake names dynamic service composition “agent based workflow services” [16]. He refers to agents as concepts, not as e.g. the FIPA agents as a means of realization. Also, the term workflow in this work does not refer to workflow management notions of the WfMC.

In the sense of general distributed workflow management, a lot of related work exists, even agent based or using Petri nets. None of these aims at providing process support to open agent networks, however, parallel development of similar ideas occurred as the field is quickly evolving: (a) Van der Aalst tried to handle complexity through inheritance: [17,18] (b) Blake’s work with respect to inter-organizational workflows [19] is related to ours, but focuses on automatic configuration and management of low-level services; (c) Buhler and Vidal cover adaptive and distributed workflows: [20,21]; (d) the CrossFlow European project [22,23], its main point is to outsource parts of a process and to connect several WFMS for this purpose; (e) Purvis et al. developed Jbees [24], using Web Services [25], agents and Petri nets. This approach is similar to ours. However, it does not support the intensive use of agent concepts as middleware technology for the implementation. Purvis in [26] does the same steps we published in [27,2]; transfer the workflow terminology to agent terminology (f) ADEPT is a general flexible distributed workflow control (see [28,29]), not aimed at intra-agent process control and not aimed at a general service for agents in open environments. ADEPT is designed to combine dynamic WF changes with a distributed execution of workflows, taking into account performance issues: [30]. Bauer, Reichert and Dadam also work on distributed WFMS under the aspect of load balancing [31]. (g) Also related to our work is the work of M.P. Singh et al. [32,33,34]. One important point in their work are long running workflows.

2.3 Agent Networks

Since early stages of standardization for agent communication through FIPA, open test environments were provided. To facilitate the vision of the possibilities of general open agent environments as a competition technique to Web Services and conceptually on a higher level than these, the European project Agentcities was created. Agentcities provides an open agent testbed [35] for agent based services and cooperations. A central node provides directory services for platforms, agents and services. The successor openNet [36] introduces a hierarchical and scalable network approach. Both agent networks introduce special agents that make up the network as such.

Infrastructure An infrastructure in the sense of our work consists of (a) technical aspects with a certain quality-of-service, (b) preconditions for participants,
and (c) a sufficient amount of participants because the infrastructural services are not useful as such, but become useful through usage. The process infrastructure for open agent networks as far as developed here focuses on point a: enactment services for system spanning processes as well as monitoring and explicit representation of processes, enabling a holistic view on spanning processes.

**Infrastructure in Agent Networks** We now consider the infrastructure that could result from a synthesis of Agentcities and openNet: agents and agent-based services are supported by providing technical services with a certain quality (such as servers for directory services under fix addresses) under certain preconditions (such as specifications on how to connect and what agents to provide), and by getting together a certain amount of users. The explicit support of spanning processes as described above is missing. As an existing support for processes could the standard interaction protocol specifications of FIPA be seen, but apart from that there is no support for application specific processes.

### 3 Conceptual Scope

This section introduces the basic ideas and concepts before in the next sections a prototype is designed. Some technical aspects are mentioned here since tools generally have an impact on the ideas (compare notion of a Think Tool in [37]). Furthermore, this section spans a frame of possible implementations. Sect. 4 chooses one possibility which is refined, until it reaches the concrete prototype described in Sect. 5.

To summarize, using the introduced concepts and techniques, we design a process infrastructure for open agent networks using Mulan as a conceptual view on agent systems and Capa as an agent platform. We use Petri nets to define the behavior of agents as well as for the definition of the precedence relation to workflows. Parts of this work and the development over the years was published before in [3,1,2].

#### 3.1 Framework Capa

The design, modeling and implementation will be done with Renew as an integrated development environment with editors for different diagram types, some code generation possibilities, and net component plugins.

Capa is the FIPA compliant implementation of Mulan realized in Renew. The complex application that we want to implement with Capa agents made it necessary to integrate additional decision capabilities in the Capa standard agent while keeping the clean concepts of Mulan agents. This resulted in an interface for decision components as reference nets to interact with the agent’s view of an interaction (protocol reference nets), to use the agent’s knowledge base, and to technically connect external tools (e.g. legacy systems or a GUI) to an agent. The main parts of the implementation of a Capa agent are to implement protocol nets and decision components and to define an ontology (Java classes are generated).
The decision component and its interface is one aspect of intra-agent communication for Capa agents. One special situation for inter-agent communication is when one agent creates another agent. The creator is the “ordering customer” of the created agent and therefore, the creator is specially trusted. Until now, in Capa no special communication channels for this situation exist, so we designed an application specific interaction for this purpose instead of using features of the framework.

In order to implement the prototype, Capa was extended to join Agentcities [38] and to openNet [39]. In order to realize the concept agent-implemented-by-agents, ongoing work aims at nesting Capa agents one into the other.

3.2 Design Approach

The design is done with the approach “Petri net based agent oriented software engineering” (PAOSE, described in [40] and [41]). This approach uses mainly the following diagrams: use case diagrams and agent interaction protocol diagrams (AIP) from (A)UML, and multi-agent diagrams based on class diagrams. All diagram types are supported as drawing plugins within Renew, the AIP plugin allows Petri net code generation and roundtrip engineering.

3.3 WFMS and Agents

We can define five levels of combination of agents and workflow management systems (WFMS): (1.) Pure forms: either WFMS or agent system. (2.) WFMS implemented using agents (e.g., instead of objects). To the outside, no difference to level 1 is visible. (3.) A multi-agent system providing the functionality of a WFMS, so the agents are visible and the functionality is used by agents. (4.) A process infrastructure can provide WFMS functionality to a specially designed multi-agent system (MAS). This introduces a new view on MAS: each process is described using workflow concepts, there are not only protocol nets describing the agent view of an interaction, but also the process management: for changes, observation, instantiation etc. of processes. (5.) Workflow management services are as ubiquitous as a transport service. Agents can use these concepts also for internal process management.

3.4 Vision: Process Control Within Agents

An arbitrary agent bears part of the overall process control within itself. All such parts of process control together form a global workflow management system (in analogy, all sending and receiving facilities within each agent in a MAS form the overall message transport system). To reach this in the long run, we aim at the transfer of the process control between agents to the process control within an agent. Such a generic process control would change the communication, coordination, and cooperation of agents in open networks.
3.5 Distribution of Agents and Processes

We transfer the basic idea of nested structures to agents: one agent is implemented using several agents which are internal agents. The overall agent acts as a platform for the internal agents. The first step to this viewpoint is to consider the protocol nets as agents, communicating with other protocol nets via the previously mentioned decision components, using the knowledge base as a common resource. Another viewpoint is to regard a multi-agent system as one distributedly implemented agent, either with or even without an explicit representation of that single agent. Stockheim et al. present a holonic way to structure multi-agent systems using one agent that represents a multi-agent system: [12]. Our approach reaches further by regarding the constituting parts as within the representative part.

In this idea, one workflow management system (WFMS) agent is made up of several specialized agents, and one level higher, several WFMS agents can be combined to implement one distributed WFMS agent. In an open agent network, the directory services would be used for dynamic addressing of distributedly implemented WFMS agents.

A process as such may be distributed through fragmentation. Fragmentation of Petri nets for workflows has been elaborated in previous work: [2,3]. Workflow fragment nets as introduced in the cited work are place-bordered Petri nets, arranged in star shape with a control Petri net which represents the overall workflow. Conflicts can be handled using a distributed lock mechanism.

4 Process Infrastructure

The following concepts are developed in several steps. Step 1 is the design of an agent based workflow management system corresponding to level 3 in Sect. 3.3. The following steps aim towards the process infrastructure (level 4). Granularity of concepts for the different steps is varying.

A process infrastructure for agent networks combines concepts known from workflow technology with concepts such as autonomy from agent technology. Using the concepts introduced in this section, a process infrastructure can be developed to control / monitor MAS.

4.1 Step 1: An Agent-Based Workflow Management System

For the first-step workflow management system, the following three types of roles were identified: (1) agent roles: e.g. workflow management system (WFMS), workflow engine (WFE), workflow enactment service (WFES), Accountmanager, database agents for tasks, workflows; (2) user roles refine the agent role “user”, e.g. administrator or task executor; and (3) application specific roles are refinements of user role “task executor”. WFMS is an agent that conceptually contains the constituting agents WFE, WFES, etc. (compare 1).
The design process of interactions resulted in interactions for session handling, subscription with WFMS for up-to-date workitem and activity lists, instantiation and finishing of workflows, and interactions to handle workitems and activities.

The ontology contains simple concepts such as “workitem” as well as predicates to make statements like “current workitems of”. Agent actions like “instantiate workflow” are special concepts that are used to define request messages.

4.2 Step 2: The Workflow Agent

In the second step, additional flexibility is added to the system by introducing an agent that represents a workflow as such. This workflow agent represents an additional indirection and a flexible connection between the creation (instantiation), the execution and changes to a workflow instance: The workflow agent knows the circumstances of creation (instantiation) including information about the requestor, the home management system, and the workflow definition. The workflow agent also knows the circumstances of execution. The workflow agent can be given extra constraints besides the mere execution as well as extra degrees of freedom regarding changes. When a change is to be made, or a decision to be taken about the future execution, all this information can be used by the workflow agent.

To distribute the execution of a workflow, the workflow agent may be implemented through various workflow fragment agents (using star shape as mentioned in paragraph Distribution of Agents and Processes). A design decision for the prototype is to let the local workflow enactment service (WFES) create a workflow agent instead of, e.g. the external user creating a workflow agent to act on the user’s behalf. This way, the workflow agent can be seen as a trusted part of the WFMS.

A workflow agent makes the interaction of WFES and workflow engine (WFE) agents more flexible. Possibilities to reassign the work of executing a workflow (meaning mainly the conflict-solving) include: (a) Workflow agent just transports the workflow definition and gives it to the local WFE for execution (this we choose as a prototype design decision). (b) Workflow agent transports and executes the workflow definition, and interacts with the local WFES as a bridge to the local WFMS. Intermediate positions are possible with different granularities, such as the workflow agent giving tasks one-by-one to the local WFE.

The workflow agent is equipped with initial information which cannot be changed from the outside after creation (such as a workflow definition and creator address). The different tasks of the workflow agent (finding an appropriate WFMS to execute the workflow, traveling, execution and finishing the workflow and finally going back to the creation place) are coordinated in a life cycle which can either be realized as a protocol net or as a decision component of a Capa agent.

1 Changes on the workflow at run-time depend on the abilities of the workflow agent.
   If it incorporates the necessary functionality, the workflow in its narrower meaning can be changed by the workflow agent itself.
4.3 Workflow Management for Open Agent Networks

As described in Sect. 2, open agent networks provide technology for an infrastructure where directory services, identity services, and message transport services are combined and provided.

A process infrastructure as developed here provides a workflow view on distributed processes in an open network. A workflow view includes views on the current state of running processes as well as on planned and past processes (logging). In a process infrastructure as envisioned here, each agent can use a generic process control for its internal processes, especially but not restricted to the case that the implementation of an agent is again distributed.

This can be reached by building a hierarchical MAS where the super-MAS implements a distributed WFMS and each simple MAS implements a local WFMS. The workflow views of the subordinate ones are culminated in the super-MAS. An agent that wants to use a WFMS needs to have access to an (open) agent network, and needs to be aware of the interaction protocols the WFMS prescribes.

5 Realization

This section introduces the two steps of the realized prototypes as well as four examples in order to illustrate the concepts detailed in the paper.

Two prototypes have been developed according to the concepts presented in Section 4.

5.1 Step 1: An Agent-Based Workflow Management System

To summarize, the step-1 prototype can manage users, roles, task definitions, form specifications, sub workflows, and rules to combine roles and tasks to rights. Users (Executors or Administrators) can log in and change configurations or execute tasks in running workflows. This means, that the basic usage scenario of a WFMS is supported through step 1. This confirms the concepts and assumptions from previous sections. This agent WFMS is qualitatively different from other WFMS because the underlying concepts and techniques are especially flexible and powerful.

5.2 Step 2: The Workflow Agent

In the step-2 prototype, the WFES creates a workflow agent upon workflow instantiation. The prototypical lifecycle is shown in Fig. 2. It consists of five states of the Workflow Agent: created, arrived (pending), pro-active, workflow execution, done. The state pro-active represents the active decision component of

---

2 A note on the syntax in the figure: places with equal names represent the same place: they are merged; edges to status places are interpreted as “is dependent on”; edges with double/no arrows do/do not modify values on status places.
5.3 Application Examples

This section will give four examples in order to illustrate the concepts detailed in the paper. The order of the examples represents the development over time.

The Phoneshop [12] is an application that is implemented as a workflow application in the RENEW workflow engine. It offers possibilities of control and monitoring on the process level and allows to gather structured and unstructured data through task-forms, manual tasks, triggering of external application and automatic execution of tasks as subworkflows. Forms, tasks and roles of participants and rules for execution are defined in a data base. Workflows are defined as reference workflow nets. These introduce the concept of a task tran-

Figure 1. Coarse Lifecycle of the Workflow Agent
sition into the Petri net formalism, which allows for notification of executors, cancellation and confirmation of task execution.

**The Agent Based Settler Game** serves as a complex example for the design and implementation of an application with distributed processes using agent technology \[38\]. Each interaction protocol (IP) is implemented as multiple protocol reference nets (one for each participant of the IP) so that the overall process and decision taking is completely distributed. While very powerful concepts and approaches were used for designing this application, the overall workflows were only implicitly modeled through the entirety of all agent behaviors, e.g. IP and decision components.

**The Travel Agency** \[44\] is based on the agent based workflow management system prototype introduced in Sect. 5. It prototypically shows that the concepts of multi-agent applications, which have a strong process-oriented character, and the concepts from workflow systems can be combined. Agent applications such as the Settler Game offer flexibility, while workflow applications such as the Phoneshop have a clear process-oriented design. In the Travel Agency, processes have been designed to offer both, clearness and flexibility.

**The Change Request Management** \[45\] uses the envisioned process infrastructure. It is a complex application and is executed distributedly on a network of agent platforms that offers workflow management functionality (such as workflow engines or workflow enactment services). Workflow agents can find services on the network, migrate to these platforms and enact and execute workflows, workflow fragments or tasks.

6 Conclusion

Building on our previous work, which introduced a multi-multi-agent systems architecture enriched with a WFMS-like interface, in this paper we present the central modelling, implementation, and prototyping ideas and concepts for a specific instantiation of such an architecture.

On the conceptual level the result of this paper is the presentation of a specific vision of a process infrastructure for open agent networks. It summarizes and gives an overview over a research project that lasted three years. To obtain the overview, all technical details were omitted. Compared to the previous publications, this paper shapes the term process infrastructure regarding the specific vision, elaborates the workflow agent, and makes the central modeling ideas explicit.

On the technical level the result of this project is a model and a prototype of an agent based workflow management system with an especially flexible architecture and a mobile agent representation of instantiated workflows. The underlying agent framework CAPA was connected to the open agent network
openNet, thus realizing step 3 and parts of step 4 towards the vision of the process infrastructure (step 5).

As the overall structuring concepts agents and multi-agent systems are used. The intensive use of abstraction and hierarchical, but flexible structuring of such multi-agent systems allows to implement a process infrastructure, which encompasses the necessary features to cope with the requirements to conceptually cover distributed, autonomous, concurrent, and complex applications. Therefore, we presented the notion of a process infrastructure for open agent networks.

The main advantage of our process infrastructure is that it is based on homogeneous and powerful techniques mentioned above. This allows for the implementation of a powerful system for the controlling and monitoring of complex applications. An important component is the workflow agent, which provides flexibility and autonomy of processes. In which way this is also applicable in the area of ULS (Ultra Large Systems) is subject to future research.

References
