Dynamic Process Integration Framework: Toward Efficient Information Processing in Complex Distributed Systems

Gregor Pavlin and Michiel Kamermans Thales Nederland B.V., D-CIS Lab Delft, Nederland
E-mail: gregor.pavlin@d-cis.nl, michiel.kamermans@d-cis.nl

Mihnea Scafeş
University of Craiova, Romania
scafes_mihnea@software.ucv.ro

Keywords: service oriented architectures, distributed reasoning, negotiation, multi agent systems

Received: February 22, 2010

The Dynamic Process Integration Framework (DPIF) is a service oriented approach which supports efficient creation of distributed systems for collaborative reasoning. The DPIF is relevant for an important class of contemporary applications requiring efficient and reliable processing of large quantities of heterogeneous information. An example of such an application is situation assessment in complex decision making processes in dynamic environments. The DPIF supports (i) a systematic encapsulation of heterogeneous processes and (ii) negotiation-based self configuration mechanisms which automate creation of meaningful workflows implementing complex collaborative reasoning processes. The resulting systems support processing based on rich domain knowledge while, at the same time, the collaboration between heterogeneous services requires minimal ontological commitments.

Povzetek: Opisan je nov pristop v procesiranju informacij v kompleksnih porazdeljenih sistemih.

1 Introduction

This paper introduces a service oriented architecture supporting complex collaborative processing in distributed systems. The presented approach is relevant for many contemporary applications that require reasoning about complex processes and phenomena in real world domains. For example, in crisis management advanced information processing is required for (i) identification of critical situations, (ii) impact assessment which takes into account possible evolution of physical processes, (iii) planning and evaluation of countermeasures and (iv) decision making. This can be achieved only through adequate processing of large quantities of very heterogeneous information, based on rich expertise about different aspects of the physical world. Such processing requirements typically exceed the cognitive capabilities of a single human expert; an expert typically does not have knowledge of all the relevant mechanisms in the domain and cannot process the huge amounts of available information. On the other hand, full automation of decision making processes in such settings is not feasible, since the creation of the required domain models as well as the inference are intractable problems. Specifically, automated inference processes involve many variables and relations with accompanying representation and inference mechanisms.

Such settings require collaborative processing based on a combination of automated reasoning processes and cognitive capabilities of multiple human experts, each contributing specific expertise and processing resources. Key to effective combination of human-based expertise and automated reasoning processes is a framework which allows that each piece of the relevant information is adequately considered in the final processing outcome. The main elements of such a framework are:

1. Standardized formats that facilitate sharing of heterogeneous information.
2. Filtering services which provide stakeholders in a decision making process with the right information at the right moment in time. In principle, filtering services must transform heterogeneous data to more abstract information types and route the information to the consumers who can make use of it.

In this paper we focus on the second element, which is tackled with the help of the Dynamic Process Integration Framework (DPIF). The DPIF supports seamless integration of heterogeneous domain knowledge and processing capabilities into coherent collaborative processes. Processes are encapsulated by software agents, each using identical communication and collaboration mechanisms. The DPIF combines Multi Agent Systems (MAS) and a service oriented paradigm in new ways which facilitate implementation of hybrid collaborative reasoning systems with emergent problem solving capabilities. In contrast to traditional MAS approaches [29, 11], the DPIF facilitates integration of human cognitive capabilities right into problem

solving processes in workflows; humans are not mere users of an automated system, but contribute the processing resources. From the problem solving perspective, the humans can be viewed as a specific type of processing modules, integrated into the overall processing system via assistant agents.

In general, a key to efficient collaborative processing in complex domains are *workflows*, in which peers with different processing capabilities exchange relevant information [25, 4, 2, 3, 27]. Often such workflows are created through dynamic composition of services [5, 25]. In this way the systems can adapt at runtime and deliver tailored solutions to specific problems. Creation of workflows is often based on centralized planning and ontologies describing relations between different services. Approaches exploiting centralized service composition have been successfully used in many relevant applications, such as business process modeling [15, 22], scientific querying [4], planning/booking systems [22] and simulation and scientific grid computing [14, 8].

For the challenges addressed in this paper, however, centralized approaches to composition of workflows and central description of relations between services are neither practical nor necessary. Namely, we are dealing with systems in which artificial agents and experts collaboratively process large quantities of heterogeneous information. In such settings construction of centralized ontologies describing services as well as all relations between the handled information types is likely to be very hard or even intractable. Similarly, centralized construction of workflows might not be practical, since the constellations of available services (i.e. automated processes or experts) change frequently at runtime. Given the time constraints, communication of all changes and system states to a central workflow composition process might not be feasible. Thus, the resulting, centrally composed workflows are likely to incorporate only a subset of all services relevant for a problem at hand.

It turns out that efficient solutions to service composition can be obtained if we explicitly take into account the characteristics of the problem. In particular, many of the challenges associated with centralized approaches to service composition and definition can be avoided if the resulting systems are used in organizations that can be characterized as Professional Bureaucracy [26]. In such organizations the skills are standardized, the control is decentralized to a great extent and the experts and/or automated processes do not have to share domain and processing knowledge. In other words, complex problems can be efficiently solved with the help of systems of *loosely coupled* experts and automated processes without a centralized control. Collaboration in such systems can be achieved through service discovery based on local domain knowledge. Therefore, we introduce an approach which does not require centralized service ontologies and centralized service composition methods. Moreover, fully decentralized configuration of meaningful processing workflows can be achieved by using *local knowledge of relations* between different services. The approach requires simple ontologies which serve primarily for the alignment of the semantics and syntax of messages exchanged between the processes in workflows. In addition, the relations between types of services are captured by local functions, dispersed throughout a system of modules providing the services. We show that meaningful workflows supporting globally coherent processing can be created by using only local domain knowledge. In this way we obtain systems which support processing based on rich domain knowledge while, at the same time, the collaboration between heterogeneous services requires minimal ontological commitments [9].

Overall, by using the DPIF encapsulation techniques and methods, arbitrary processing services can easily be made composable and negotiable.

The paper is organized as follows. In section 2 a rationale for decentralized collaborative reasoning in workflows is provided and the basic features of the DPIF are introduced. Section 3 explains how meaningful workflows between heterogeneous processes can be dynamically implemented, without centralized knowledge of relations between the variables in the reasoning processes. In particular, we emphasize a combination of service composition, decentralized validation methods and advanced negotiation mechanisms, which allow a systematic incorporation of various criteria into the workflow creation processes. Section 4 introduces basic DPIF architecture principles while section 5 introduces an approach to efficient construction of service ontologies by exploiting the local domain knowledge captured by different DPIF agents. Section 6 provides conclusions and plans for the future work.

2 Collaborative processing

Reasoning about domains requires knowledge about typical dependencies (i.e. relations) between relevant phenomena in these domains. By using (i) domain models capturing the relations between relevant phenomena and (ii) evidence based on observations of certain phenomena, we can assess (i.e. estimate) the states of the domain that cannot be observed directly. In addition, with the help of models, the future evolution in domains can be predicted. However, in complex domains reliable reasoning can be achieved only by relating large quantities of information of very heterogeneous types with very different semantics. Such dependencies can be explained only through complex models.

Irrespectively of the used models, it is unlikely that in complex domains a single model designer or an expert understands all the relevant phenomena and relations between them. Instead, a set of relatively simple domain models will exist, with each model capturing a small subset of the relevant variables and the corresponding relations. Thus, reasoning based on the relations between the entire available evidence can be achieved only by combining simpler processes, each using a limited domain model. The out-

puts of simple processes are used as inputs of other simple processes. In other words, the reasoning is based on data-driven workflows established between heterogeneous processes. In such workflows difficult problems can be solved through collaboration of heterogeneous processes, each focusing on a relatively small subset of relevant aspects in the targeted domain.

We illustrate such processing by using an example from the environmental management domain. In a chemical incident a leaking chemical starts burning which results in harmful fumes. The impact of a resulting fumes is mitigated through a collaboration of experts captured by figure 1. We assume that the factory staff (FS) at the incident have an overview of the current state of the damaged system; FS can estimate the quantity of the escaping chemical and its type. This information can be used by a chemical expert at the incident location (CE1) to estimate the type and quantity of toxic fumes resulting from the fire. By knowing the location of the fire, the meteorological conditions, and the quantity and type of the produced fumes, chemical expert (CE2) can estimate the zones in which the concentration of the toxic gases have exceeded critical levels and identify areas which are likely to be critical after a certain period of time. The CE2 makes use of the domain knowledge about the physical properties of the gases and their propagation mechanisms. In addition, it guides fire fighter teams (MT) which can measure gas concentrations at specific locations in order to provide feedback for a more accurate estimation of the critical area. A map showing the critical area is supplied to a health expert (HE) who uses the information on population obtained from the municipality to estimate the impact of the toxic fumes on the human population in case of exposure. Finally, the estimated impact on the population is supplied to decision makers, who choose between no action, evacuation and sheltering. This decision also considers estimated time and costs in case of an evacuation of people from the danger zone as well as the estimated costs and duration of a preventive evacuation. The former estimate is provided by the fire brigade representatives while the latter estimate is supplied by the police department. In other words, in such a system, each expert can be viewed as a module providing predefined services which in turn require services from other experts. Thus, the situation analysis in the presented example can be viewed as a workflow between different, weakly coupled processing services, each specialized in specific aspects of the domain. Moreover, a processing service can be provided by a human (e.g. a chemical expert analyzing the extent of the contamination) or by an automated reasoning process (e.g. detection of gases based on automatic fusion of sensor data). Note that, for the sake of clarity, the used example is a significant abstraction of real crisis management processes.

Moreover, the example can be seen as a class of problems where we have to reason about a situation which can be viewed as a specific combination of known types of events and processes, each understood by a human expert or modeled by an artificial agent. For example, the way chemicals burn and react, the effects of exposure to toxic fumes, evacuation approaches in hospitals and schools, etc. are independent of the location and time. Therefore, we can obtain general knowledge about such processes and phenomena which can be used for the analysis in any situation involving such phenomena. In other words, a mapping between experts and artificial agents, on the one hand, and event types, on the other hand, can be made a priori; we can assign roles to different experts and artificial agents based on their domain knowledge and models.

Since each situation (e.g. chemical incident) is a unique combination of known types of events, a specific workflow consisting of a particular combination of processing nodes is required for adequate situation assessment. In addition, due to unpredictable sequences of events it is impossible to specify an adequate workflow a priori. For example, given the wind direction, experts for the evacuation of hospitals and schools might be needed. However, if the gas is blown to the open sea instead, no evacuation experts are needed in the situation assessment process.

Clearly, a major challenge is creation of adequate workflows which correctly integrate the relevant processes and support globally coherent processing in decentralized collaborative systems. In the following text we explain how this can be achieved in an efficient and sound way.

2.1 Dynamic process integration framework

The Dynamic Process Integration Framework (DPIF) supports decentralized creation of workflows that facilitate collaborative problem solving. The DPIF is a service-oriented approach (SOA) which supports efficient composition of very heterogeneous processing services provided by different experts and automated reasoning processes. In the context of the DPIF, the information processing is abstracted from human or machine instances; a reasoning process is either provided by a human expert or an automated system implemented by a software agent. Each process provides a well defined reasoning service in the form of an estimate, prediction, cost estimate, etc. The inputs for each of such processes are provided by other processes or by direct observations (i.e. sensor measurements and reports from humans).

A human expert or an automated inference process is represented in the system by a software agent, a functional (i.e. processing) module which (i) supports standardized collaboration protocols and (ii) allows incorporation of arbitrary reasoning approaches. In other words, the agents provide a uniform communication/collaboration infrastructure allowing seamless combination of heterogeneous processes provided by human experts or implemented through AI techniques. Each agent registers in the DPIF-based system (i) the services supported by its local processing capabilities and (ii) the required inputs, i.e. types of services that should be provided by other agents in the system.

By using the registered services, agents distributed

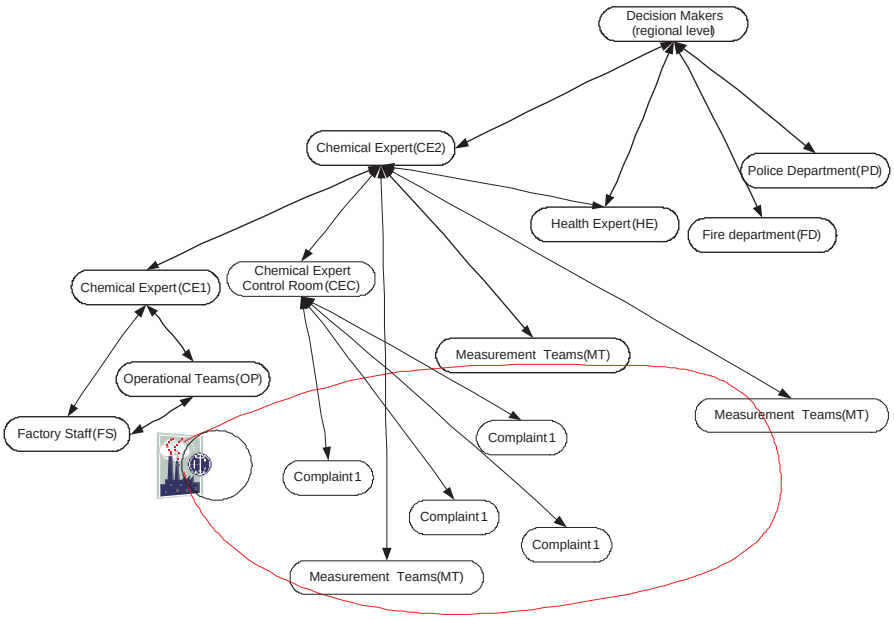

Figure 1: A workflow in a decision making process. Arrows denote information flow between different experts, each processing relevant information of different types. The circled region denotes the initial estimate of the area where concentration is likely to be critical.

throughout different networked devices can autonomously form workflows in which heterogeneous processes introduce collaborative reasoning. The configuration of workflows is based on the relations between services captured by local models; each agent *knows* what service it can provide and what it needs to do this. This local knowledge is captured by the relations between the variables in partial domain models. Thus, *no centralized ontology describing relations* between different services of various agents is required, the creation of which is likely to be intractable.

In other words, globally coherent collaborative processing is possible by combining local processes, without any global description of relations between inputs and outputs.

In the following discussion we focus on (i) principles for the creation of valid workflows based on the local processing capabilities of different agents and (ii) describe the basic elements of the DPIF architecture.

3 Processing workflows

A basic workflow element in the DPIF is a local process. Moreover, in the following discussion the term local process refers to a reasoning process provided either by a human expert or an automated system implemented by a software agent. Each local process corresponds to a function $F : \{X_1, ..., X_n\} \rightarrow Y$, mapping values in a domain $\{X_1, ..., X_n\}$ to values of some variable of interest Y. The value of Y for particular values of arguments is given by $y = f_y(x_1, ..., x_n)$.

Such functions can be either explicit, based on some rigorous theory, or implicit, when they are provided by humans or sub-symbolic processes, such as for example neural networks. An example of a mathematically rigorous mapping is the function $x_{CE1} = f_{x_{CE1}}(x_{FS})$, an explicit formula describing the relations between the fume volume per time unit represented by X_{CE1} and the escape rate of chemicals denoted by X_{FS}. This function is used by the Chemical Expert CE1 in figure 1. An implicit mapping, on the other hand, is performed by the health expert (HE) who estimates the critical regions with respect to the impact on the residents. HE interprets information about critical concentration X_{CE2} in combination with information on population distribution X_{POP} by using an implicit function $x_{HE} = f_{x_{HE}}(x_{CE2}, x_{POP})$.

3.1 From local to global processes

An expert or an artificial agent often cannot observe values of certain variables; i.e. variables cannot be instantiated. Instead, the inputs to the local function are supplied by other processes forming a collaborative workflow (see section 2). Thus, the inputs to one function are outputs of other functions used by the information suppliers. From a global perspective this can be seen as a function composition; in a function, each variable which cannot be instantiated is replaced by a function. This process continues until a function is obtained in which all variables are instantiated, i.e. all free variables in the resulting nested function have been reduced to direct observations. In this way, a global function emerges as different processes are connected in a workflow. The resulting function is a com-

posite mapping between directly observable variable states and hidden variables of interest.

In other words, a workflow in a DPIF system corresponds to a full composition of functions, in which each variable replaced by a function corresponds to a required service. This yields the value of the variable of interest. Let's assume an example with six service suppliers shown in figure 2(a), using the following functions:

$$\begin{aligned} x_a &= f_a(x_b, x_c), x_b = f_b(x_d), x_c = f_c(x_e, x_f), \\ x_d &= f_d(x_g), x_e = f_e(x_h), x_f = f_f(x_i). \end{aligned}$$

then the workflow supporting collaborative computation of the value for x_a corresponds to the composite function

$$f_a(f_b(f_d(x_g)), f_c(f_e(x_h), f_f(x_i))) \quad (1)$$

It is important to bear in mind that in DPIF no explicit function composition takes place in any of the agents. Instead, the sharing of function outputs in a workflow corresponds to such a composite function; i.e. a workflow models a (globally emergent) function, mapping all observations of the phenomena of interest (i.e. evidence) to a description of some unknown state of interest.

Each workflow corresponds to a system of systems, in which exclusively local processing leads to a globally emergent behavior that is equivalent to processing the fully composed mapping from direct observations to the state of the variable of interest.

3.2 Decentralized validation of workflow structures

The functions in a workflow can be described through different ad-hoc or theoretically sound frameworks (such as for example Bayesian reasoning). However, the relation between workflows and function composition has several interesting properties regardless of the theory used to describe the functions. Workflows, in general, can be represented by directed graphs[1] whose topologies have important implications regarding the reasoning (e.g. see figure 2(b)). Particularly important concepts in graphs are loops and cycles. Loops occur when there is more than one way to travel between two nodes by following the directed links in a graph, i.e. there exist multiple directed paths between two nodes (e.g. see figure 3(b)). Cycles occur if a node can be revisited by following a directed path (see figure 4(c)).

Loops and cycles provide an important insight into the dependencies between different services (i.e. processes) and thus the use of information in a distributed system.

If a process uses multiple inputs obtained from services belonging to a *loop*, then these inputs might have been generated by distributed processes using the same information. In other words, these inputs may not be independent, which might lead to data incest [7], if the inputs are not properly treated. In case of data incest, the same information is reused multiple times which is likely to lead to misleading conclusions. In case of rigorous reasoning approaches, such as Bayesian networks, this problem can be avoided by clustering of variables [12], which allows correct probabilistic reasoning even if the graphs contain loops. In any case, we should be aware of loops in systems implementing distributed functions.

While loops may be permissible, as there are various ways to deal with them if detectable, *cycles* are not permitted in workflows that implement inference, as they would lead to one or more components in a workflow processing misleading data. That is, the system is generating outputs without adding new information to the system. In a data-driven approach this leads to a self perpetuating system which is likely to produce outputs which do not reflect the reality. Figure 4(b) shows an example of a system with a directed cycle, where agent A keeps supplying inputs to agent C, which in turn produces new inputs for A. By looking at the composed function represented by a workflow, cyclical workflows would lead to infinite composition sequences: if some function is directly or indirectly dependent upon itself, then this would lead to an infinitely recursive composition of the full function which is likely to result in misleading outputs.

Therefore, an integral part of the DPIF is a cycle detection mechanism based on the cycle detection principles introduced in [13]. The used approach allows a fully decentralized detection of cycles based on peer to peer communication. These principles were initially used in modular systems using Bayesian networks [13], but the technique discussed is easily carried over to a more generic domain.

In order to effect cycle-free workflows, each DPIF agent must be aware of the outputs of other agents that influence the inputs of this agent. In [13] it was shown that this information can be obtained during the creation of workflows by passing simple information between the peers in a workflow; as agents link up to form workflows, the composition of the network influencing the inputs of each agent can be recorded. By using such knowledge, agents can locally determine whether extension of a DPIF workflow with a new peer would introduce cycles; i.e. each agent can find out whether or not it is setting up an (in)direct dependency on itself, without any central authority.

While in this paper we cannot describe all interesting aspects of this approach, we illustrate the basic cycle detection principles from [13] with the help of an example. In figure 4(a), agent A supplying x_a forms a workflow with agent B supplying x_b. Agent A knows the output variable set $S_A = \{x_a, x_b\}$, and agent B knows the output variable set $S_B = \{x_b\}$. In figure 4(b), an agent C, able to supply x_c, joins the workflow. Agent C knows its current set of output variables; $S_C = \{x_c\}$. Before joining, agent A verifies whether the proposed connection does not introduce cycles. This is the case if a simple intersection test yields

[1] A directed graph representing a workflow consists of nodes, each representing a process (i.e. a function), and directed links which capture direct dependencies between the processes (i.e. supplier-consumer relations).

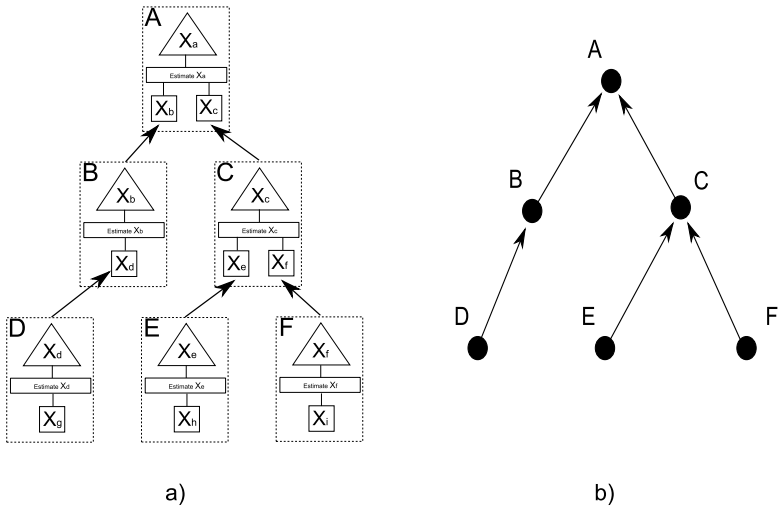

Figure 2: a) A self-organized system of agents. Each agent supplies information concerning a particular variable of interest in the domain. These outputs are based on other inferred or directly observed variables. b) A directed graph capturing the workflow between the agents from a).

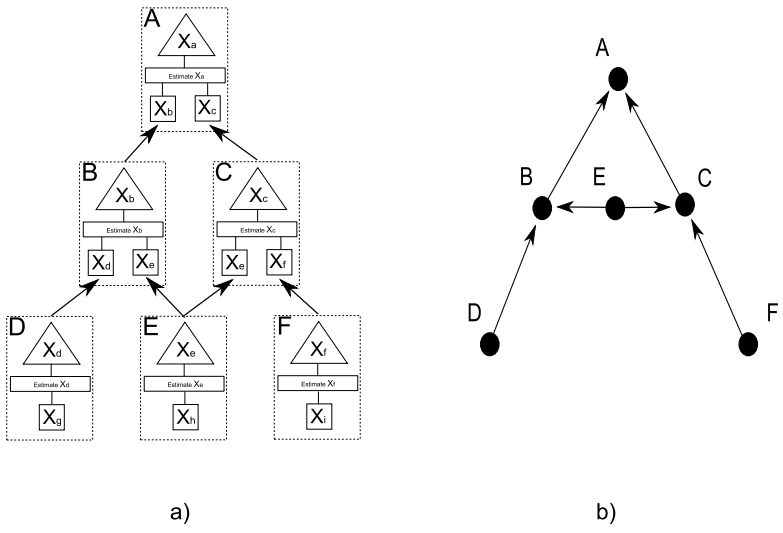

Figure 3: a) A self-organized system of agents forming a workflow corresponding to a multiply connected graph. b) A directed graph capturing the workflow between the agents from a).

$S_A \cap S_C = \emptyset$. This holds, so the connection is allowed and Agent C joins the workflow. As it does so, Agent A updates its variable set to $S_A = \{x_a, x_b, x_c\}$; i.e. A obtains all the relevant information about the services that influence its function by peer to peer sharing of local information.

However, when agent C looks for suppliers, the only available agent supplying x_a is A, the one to which C is already connected. As C conducts a verification step, in which the variable sets $S_C = \{x_c\}$ and $S_A = \{x_a, x_b, x_c\}$ are tested for empty set intersection, the intersection $S_A \cap S_C \neq \emptyset$, and so C knows that a cycle would be introduced if the service x_A were supplied to it by A.

In fact, in [13] it was shown that cycles, as well as loops, can be detected in workflows in completely decentralized manner by collaboration of peers exchanging asynchronous messages. Peers check the intersections of dynamically assembled variable sets at different levels of the workflow, and as new agents join the workflow the new network layout needs to be reflected in all agents whose downstream network has changed by new connections. Thus, we can view the task of loop and cycle detection as a combination of (i) checks which *travel* upstream (i.e. toward the top agent) until the top agent of the network is reached, (ii) messages conveying the updated topology, and (iii) control messages which lock/unlock the agents for local checks.

In general, this approach allows for an intelligent handling of loops and cycles in workflows, where the choice on whether to allow a connection or not is dependent on the function performed by an agent that is responsible for expanding a workflow. There exist functions which require that all inputs are provided, in order to yield an output. In such cases, an agent modeling a function may decide to abandon a workflow when one or more of its inputs would lead to a cycle (or loop). On the other hand, there are also functions which yield output even when some inputs are left unknown, such as for example marginal conditional probabilities expressed with the help of Bayesian networks. In these cases, an agent modeling such a function may keep participating in the workflow, provided it can ensure that the inputs otherwise responsible for introducing cycles are not supplied to any other agent; i.e. the inputs are ignored in the evaluation of the function.

3.3 Negotiation

In the DPIF, communication links between local processes in agents are facilitated firstly using service discovery: whenever an agent supplying some service (we will call this service the *parent service*, and the agent implementing it the *parent*, or *manager agent*) in a workflow requires data relating to some other service (we will call this required service the *child service*, and the agent implementing it the *child*, or *contractor agent*), a communication link needs to be established between the parent agent and the child agent. However, there are two important aspects that affect whether and why links are established: i) we might have several agents in the system that provide the same service, i.e. that are able to realize the same task, and ii) we cannot always assume that a service providing agent will automatically agree to supply the service asked for by a requesting agent. For example, the providing agent might be overloaded, or it might even consider that establishing a link is inappropriate, given the current context.

In addition, on its own, service discovery can only offer links between agents based on a broad level of service matching, while for the system to solve a particular problem, a finer level of control is required to match services on whatever additional parameters may be of importance to particular links. For this we use negotiation. Rather than performing perfect matching at the service discovery level, negotiation allows us to filter potential links found through service discovery based on additional service parameters.

Negotiation in general consists of three elements [10]:

- protocols, i.e. sets of rules that describe the steps of negotiation processes; example protocols are Contract Net (CNET), monotonic concession protocol (MCP), Rubinstein's alternating offers and auctions [28] [24] [29].
- subject, i.e. the item being negotiated about. In service negotiation, the negotiation subject is the service with its parameters.
- strategies, i.e. the set of decisions that agents will make during the negotiation in order to reach a preferred agreement.

We have developed a conceptual framework for service negotiation that will be used in the DPIF [21]. The framework is generic and addresses negotiation protocols, negotiation subject and decision components (how agents make proposals and how they select the best proposals).

Establishing links is based on one-to-many negotiation; i.e. one agent (the manager) negotiates with multiple agents (possible contractors) about a service, with an arbitrary set of parameters (multi-issue subject) [19], [20]. The framework defines the common steps of negotiations, including starting negotiations, making proposals, deciding whether an agreement or a conflict has been reached and termination.

The negotiation subject consists of the service plus a subset of service parameters that are important decision factors during negotiation (they are taken into consideration when selecting the proper service providers). During negotiation, these parameters are considered negotiation issues. Thus, the negotiation designer defines the negotiable parameters of the service (negotiation issues) when configuring the service. When negotiating, agents need to know how to handle the issues and extra information about the issues should be added into the system. Therefore, issues have a set of properties. These properties include:

- *name*: a unique identifier of the issue in the subject.
- *data type*: the type of the value the issue is allowed to take.

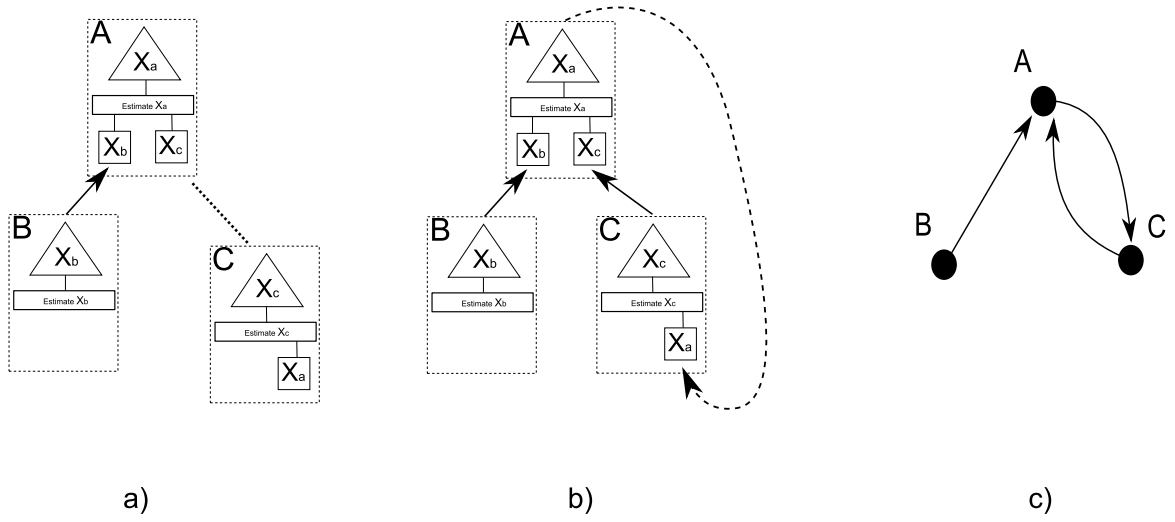

Figure 4: a) a partially formed collaboration, allowing the top agent to perform inference, leading to information about the state of variable x_a. b) If the potential connection represented by dashed directed link were allowed, a cycle would form in the workflow. c) A graph corresponding to a system of DPIF agent if the connection from b) were allowed.

– *value type*: specifies whether the value of the issue can be modified in proposals or the original value set by the manager should be kept.

In addition to the standard set of properties, agents (depending on their role) are allowed to assign some extra properties to the issues:

– *weight*: represents the relative importance of the issue in the subject, compared to the other issues.

– *reference value*: represents the ideal value of the issue from the point of view of the agent.

For the moment the issues are evaluated independently.

The default negotiation in DPIF is an implementation of CNET [24]. A manager starts a negotiation for a service by sending a call for proposals to all the contractors that are able to provide the service. Each contractor then tries to compute a proposal. Contractors evaluate possible proposals using a utility function that takes into consideration the effort necessary to provide the service under the conditions in the proposals. If contractors are successful in generating proposals, they send them to the manager, otherwise they withdraw from the negotiation. After receiving proposals, the manager selects the best proposals, the ones that give him the highest utility. The manager uses a weighted sum utility function to evaluate proposals.

Although the default negotiation protocol is CNET, the system allows for the use of arbitrary negotiation schemes, supporting domain specific one-to-many generic negotiation approaches where the protocol, the subject type and strategies are specified in the service configuration phase. There are two configuration levels:

– *negotiation type level*. At this level the negotiation designer defines the negotiation protocol, identifies the negotiation issues and sets default values for their properties.

– *negotiation instance level*. At this level a human expert can tune the issue properties for a certain situation.

We illustrate how negotiation takes place with such multi-issue subjects, by looking at the example described in figure 1. We emphasize the step when the CE2 decides to guide MTs to measure gas concentrations at a location *X*. In addition, CE2 decides that measurement devices *DEV_X* are the most appropriate for the measurements. Other devices can be used as well but with less precision. CE2 initiates a negotiation over the multi-issue subject (*Gas measurement, location, device*) with all MTs that are able to provide the service *Gas measurement*. MTs propose various deals for this negotiation: the locations they are currently placed at and the devices they have. Ideally, MTs would propose the best deal (*Gas measurement, X, DEV_X*). CE2 must decide what MTs to choose by taking into account various parameters: the distance between location *X* and locations where MTs are placed, the differences in precision between device *DEV_X* and devices MTs have.

The outcome of a successful negotiation between two agents is a contract, which results in the creation of a communication channel supporting peer-to-peer communication between the agents.

3.4 Collaborative information acquisition

A DPIF-based system of experts and automated reasoning processes typically requires acquisition of large amounts of very heterogeneous information obtained at different locations by using complex acquisition procedures.

The DPIF approach supports efficient gathering of information by distributing the search task throughout systems of collaborating agents, i.e. services. Namely, the search for the relevant information/data must be based on the domain knowledge in the system, which is used for the reasoning; only the information that is represented in implicit or explicit domain models can be processed. Since in the DPIF the domain knowledge is dispersed throughout a system y the experts or automated processes providing the reasoning services, the information gathering is carried out by the service providers as well.

Upon establishing contracts in a particular collaboration network, the service providers initiate a search for information based on their domain knowledge. Each service provider knows exactly which information is needed and under what conditions it should be obtained. The service providers either request inputs via the DPIF service discovery or they have access to direct observations (e.g. they observe sensors, databases, etc.). We say that the services with a direct access to the observations are *grounded* in the domain; they can observe real world phenomena directly. If the services are not *grounded*, they delegate the search of information and pass on only the information which *constrains* the search process carried out by service provider. Typically, an information requester knows *what* is needed and *where* and *when* it should be obtained but does not know how this information locating can be carried out. This information is passed on to the provider, that knows *how* the information can be obtained in the specified spatio-temporal context.

In this way a complex search for information is broken down into several simple search processes which guarantee that only the relevant information is inserted into the distributed reasoning system. By using the service discovery and negotiation, the judgment of what is needed and what can be provided is carried out by the information consumers and providers themselves, without the need of introducing a central authority. Note that a centralized approach to information search would require a centralized authority that would replicate some of the local knowledge and have a complete overview of the information acquisition processes and capabilities of all services. Such a central solution is not practical in the targeted domains, where new expertise/algorithms as well as new information sources are added frequently.

We illustrate the distributed approach to information acquisition with the help of the aforementioned example. Expert CE2 is requested to estimate the critical zones after a leak at a chemical plant is discovered. CE2 needs additional information to accomplish this task. Based on his local expertise, CE2 knows *what* information should be obtained and under what conditions this should happen. For example, CE2 needs the information about the type and quantity of toxic fumes escaping at location X (assuming he was just informed that there was a leak at X). He also needs information about the weather in the larger incident area. This information should be obtained within a certain time interval and the service itself should conform to certain quality requirements, pertaining to such things as the experience of the information providers, minimum precision of the estimates, etc. The requested types of information as well as the additional constraints are used by the DPIF service discovery and negotiation mechanisms which establish contacts between the DPIF agents of the requester CE2 and suitable information providers, such as expert CE1. The negotiation ensures that the requested types of information pertain to the right location and time interval and that the experts with the right quality of service are involved. For example, a request from CE2 is processed in the DPIF, which finds agents of two experts that can provide service of type CE1. However, during the negotiation, it may turn out that only one expert can provide the estimate of the toxic fumes at location X within the given time interval. The chosen CE1 can get to the location on time and supports the required quality of the service. CE1 gets to the incident location and obtains from the factory staff the information about the leaking chemical, the pressure and the leak location; i.e. CE1 knows what information should be obtained about the incident site and how this can be done. This information is used by CE1 for the estimation of the quantities of the escaping fumes. This estimate is routed to CE2 by using the peer to peer communication channel. After the first estimate of the critical areas is provided, CE2 requests additional concentration measurements at certain locations, in order to improve the initial estimate of the critical zone. CE2 specifies the types of measurements, the measurement locations as well as the maximum time interval in which the measurements must be provided. The requests are routed via the DPIF to the measurement teams (MT) who are able to go to the specified locations and carry out concentration measurements. The subsequent negotiation results in the creation of contracts between the CE2's agent and the agents representing the MTs which can then get to the specified locations in the given time and carry out the requested measurements.

Note that in this simplified example the search of information requires very different types of requests and acquisition methods. In other words, a lot of domain knowledge as well as procedural knowledge is required.

The requests to CE1 and MT reflect CE2's knowledge of the gas propagation processes, his current knowledge of the situation as well as his processing capabilities. The bids from MT and CE1, on the other hand, reflect their capabilities to provide the requested information. As a result of this approach, from a global perspective, the information acquisition process implements very complex patterns in non trivial spatio-temporal context.

4 Processing modules: architecture

Each local process (human or machine-based) is encapsulated by a software agent. In the DPIF, agents provide a uniform interface for the collaboration between local pro-

cesses in different agents. A key feature of DPIF agents is asynchronous, data-driven processing in complex workflows. This is achieved through a combination of weakly coupled processes inside individual agents, and service based coupling of processes between distinct agents.

Each agent consists of at least two processes implemented through asynchronous threads communicating via a local blackboard (see figure 5). The "Communication Engine" process is a thread that allows for message-based inter-agent communication, facilitating collaboration and making negotiation capabilities known to other agents. Through their Communication Engines, workflows between local processes in different agents can be established, by executing service discovery and negotiation (see section 3.3). The "Processing Engine" process, on the other hand, is a thread which encapsulates arbitrary automated or human based inference processes. The Processing Engine is responsible for the local processes that implement transformations between different types of information, based on the interpretation of complex cues. Moreover, each Processing Engine can keep track of one or more of these local processes simultaneously. The Communication Engine supplies the Processing Engine with inputs that are obtained through inter-agent messaging, by posting these on the agent's local blackboard for the Processing Engine to see. The Processing Engine then places the results of local inference processes on the local blackboard for the Communication Engine to pick up and relay via normal inter-agent messaging to interested agents.

The Communication and Processing engines must be able to execute simultaneously. Reasoning can be computationally expensive which requires a certain amount of time, but during this time an agent should be able to negotiate about possible collaboration with other agents, asynchronously collect their outputs for use in local processes and so on.

Both the Communication Engine and Processing Engine threads communicate through a limited set of messages via the local blackboard. New externally received inputs are transformed and placed by the Communication Engine on the internal blackboard, which triggers callback of adequate functions in the Processing Engine. The Processing Engine transforms these inputs into local output and places this output on the blackboard, which triggers callback at the Communication Engine for relaying this information via normal messaging to agents interested in this output. Through cascaded processing in the resulting system of collaborating agents, each piece of information influences the outcome of the collective reasoning processes, such as estimates/predictions or evaluations (i.e. reasoning result); with each contributing process, new observations and a particular expertise are incorporated into the global reasoning process.

Note that, irrespective of the type of the local processes, the Communication Engine and the Processing Engine in each agent use the same mechanisms for the creation and maintenance of correct workflows between local processes in different agents. The uniformity of configuration mechanisms can be used for a seamless integration of human experts into complex processing workflows. A DPIF agent representing an expert has a very simple Processing Engine which delivers the required inputs to a human expert via a suitable GUI. The expert's conclusions, i.e. his/her service, are also formulated with the help of this GUI and routed to interested agents via the Communication Engine. Thus, agents used by human experts merely provide automated routing of information between experts, and take care of the automated creation of collaboration connections.

5 Dynamic service ontologies

In order to be able to automatically compose heterogeneous services provided by different developers or experts, the definitions of service interfaces have to be standardized, which is achieved with the help of explicit service ontologies. Moreover, the locality of the domain knowledge in the DPIF approach can be exploited for efficient creation of rigorous service descriptions.

Services declared in the DPIF are typically provided by many stakeholders from different organizations whose capabilities evolve with time. Large systems of service descriptions have to be maintained and it is very difficult to specify a complete set of services prior to the operation. In other words, traditional approaches based on rigorous centralized ontologies, such as for example [4, 22], which capture service descriptions and relationships between information provided by different types of services are not practical; we simply do not know which relevant services will be available in the future and maintenance of large ontologies is likely to be very expensive or even intractable.

Fortunately, the locality of domain knowledge in the DPIF approach supports efficient creation of service ontologies. Because self organization and processing are based on domain knowledge encoded in local functions, we can avoid traditional approaches to constructing centralized ontologies, which describe domains in terms of complex relations between the concepts corresponding to the processing services. In the targeted domains, adequate maintenance of such ontologies is likely to be an intractable challenge. Instead, the services and relations between them are described by using two types of light weight ontologies:

- *The global service ontology* merely captures service descriptions, the semantics and syntax of messages used for (i) service invocation and (ii) dissemination of service results. This ontology is used for the alignment of the semantics and syntax of service descriptions at design time.

- *Local task ontologies* coarsely describe relations between different types of services supplying different types of information. In principle, they describe which types of services provide inputs to the function used by a specific service. These relations reflect the

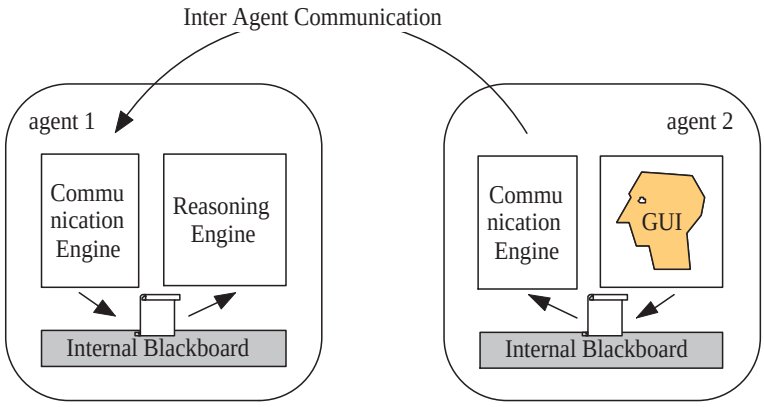

Figure 5: Interaction between agents providing heterogeneous processing services. Both agents use identical communication engine. However, agent 1 encapsulates automated processing while agent 2 introduces human-based processing.

local knowledge of each processing module. Moreover, the local ontology supports runtime creation of workflows based on service discovery.

The global ontology is a central element of the service description procedure. In order to make sure that all agents speak the same language, the global ontology captures three types of elements, namely (i) a verbal description of the service to be provided, (ii) conditions under which this service can be invoked, and (iii) a collection of representational elements resulting from the information gathered by this service. While the vocabulary with which these descriptions can be specified is rigidly formalized, it is rich enough to allow the description of arbitrarily complex services. The global ontology is used by a matching process in which service suppliers are provided with a list of existing service descriptions, based on keywords and free text. The descriptions retrieved from the global ontology are displayed in a form that facilitates inspection of the relevant subset of existing services. If an existing service description corresponds to the new service, it is adopted. Otherwise a service definition editor allows the experts to provide a new service description, which is then added to the global ontology. By making experts responsible for deciding whether they perform a role similar to another domain participant or a genuinely new role, we overcome the problem of an a priori defined ontology that is likely to be unable to account for all aspects of the domain and expert capabilities.

The *local task ontologies*, on the other hand, are created with the help of a task definition tool which supports specification of the required inputs (provided by other services) for each provided service. In this way different services are related locally, based on the local domain knowledge. The task ontologies are stored with agents of participating experts. These relations captured by local task ontologies are central to the service discovery, which is typically initiated from within the local services. Consequently, if each expert is made responsible for the description of relations between the provided and the needed services, systems using complex relations between services can be built in a collaborative way, without any centralized configuration/administration authority.

In other words, the introduced division into a global service ontology and local task ontologies dispersed throughout the system of agents allows collaborative definition of services by experts.

Construction of such ontologies is facilitated by a collection of tools that (i) help the user discover the services in the global ontology by using keywords and written language, (ii) provide an interface facilitating inspection of the human readable descriptions and (iii) editors for defining local task ontologies. By using these tools, the experts define elements of the global service ontology and the local task ontologies without using any formal language. At the same time, the tools automatically translate expert inputs to rigorous ontologies captured in the OWL format. In other words, by deploying the two types of ontologies in combination with simple construction procedures, rigorous, machine understandable service descriptions can be created without any formal knowledge of the underlying ontology techniques.

Similarly to the DPIF approach, the OpenKnowledge framework [23] avoids creation of centralized heavy weight ontologies describing all aspects of the domain. However, while the DPIF requires a mere specification of the provided and supplied services, the OpenKnowledge framework also requires specification of *interaction models* shared by the collaborating peers. Such interaction models define workflows for each processing task a priory; the OpenKnowledge approach assumes that collaborating peers understand interaction protocols and the processing sequences of collaborating peers. This can introduce additional complexity to the system configuration in which services and processes are specified. Since the DPIF is targeting Professional Bureaucracy systems [26], it is assumed that experts do not have to share knowledge about their local processes.

6 Conclusions and future work

The DPIF supports uniform encapsulation and combination of heterogeneous processing capabilities which are required for collaborative reasoning about complex domains. The processing capabilities can be provided by human experts or automated reasoning processes. In the DPIF context, human expertise and automated processes are abstracted to functions with well defined outputs and inputs; each function provides a particular reasoning service given certain inputs.

The DPIF provides *function wrappers*, software agents which standardize function interfacing. The interfaces are based on standardized service descriptions as well as uniform self-configuration, negotiation and logical routing protocols. With the help of the DPIF encapsulation methods very heterogeneous services can be made composable and negotiable.

The DPIF agents support automatic formation of workflows in which heterogeneous functions correspond to suppliers and consumers; outputs of some functions are inputs to other functions and so on. In other words, a workflow corresponds to a set of nested functions that captures dependencies between very heterogeneous variables. Creation of workflows and routing of information is based on the relations between different types of information. These relations are captured by *local functions* wrapped by different modules. The DPIF approach assumes that each expert or an automated process can declare the inputs and outputs of the contributed local functions, which is sufficient for automated creation of globally meaningful workflows by using service discovery. Thus, in contrast to traditional approaches to processing in workflows, *neither centralized configuration of workflows nor centralized knowledge of the combination or routing rules are needed*. The resulting systems support processing based on rich domain knowledge while, at the same time, the collaboration between heterogeneous services requires minimal ontological commitments.

Decentralized creation of emergent processing workflows, based on local domain knowledge and negotiation, is useful in dynamic domains, such as crisis management, where it is difficult to maintain a centralized overview of the resources. However, if applications require optimization of workflows, centralized approaches to workflow construction might be necessary. The DPIF approach supports also centralized approaches in several ways. Namely, the DPIF facilitates creation of processing modules whose services can easily be composed by centralized algorithms; the local task ontologies provide the information on the compatibility of services, i.e. possible service combinations in workflows.

In principle, arbitrary automated reasoning techniques can be integrated into the DPIF. However, globally coherent reasoning in such workflows can be achieved only by using rigorous approaches to designing local models and combining partial processing results. Globally coherent and theoretically sound collaborative reasoning is in general very challenging and it has not been discussed in this paper due to the limited space. An example of a theoretically sound collaborative inference system based on the DPIF is the Distributed Perception Networks (DPN), a modular approach to Bayesian inference [16]. The DPN is a fully automated DPIF variant that supports exact decentralized inference through sharing of partial inference results obtained by running inference processes on local Bayesian networks [18] in different collaborating DPN agents. If the local Bayesian networks are designed according to the rules introduced in [16], it can be shown that a collaboratively computed posterior distribution for any variable in the distributed system correctly captures all evidence. The DPN framework has been used for the implementation of robust distributed gas detection and leak localization systems based on Hidden Markov Models [17].

In general, however, it can be difficult to prove that a collaborative processing approach is sound (i.e. globally coherent), especially if algorithms and the used models are not based on rigorous theory, which is especially the case with human based reasoning. Recent research on structuring of human collaborative processing in DPIF-based systems indicated that some processing rigor can be introduced if the experts implement causal reasoning [6]. In such settings the reasoning could be structured with the help of qualitative models, such as causal graphs which allow exploitation of d-separation and concepts from Hidden Markov Models (HMM) [1], such as observation and dynamic process models defined over discrete time-slices. Such structuring has several advantages. Firstly, we can prevent processing that involves the so called data-incest and recycling of information in infinite, self perpetuating reasoning cycles. Secondly, we can introduce efficient control of discrete reasoning phases throughout time slices, which supports sound temporal reasoning.

In addition, several challenges remain with decentralized creation and control of workflows. It turns out that the complexity of workflow control depends on the problem itself and the used processing paradigm. In cases in which the reasoning can be structured as HMMs, valid workflows can easily be created and time-slices can be controlled via emission of simple reset messages within the system. In such settings no workflow deadlocks can occur and information can unambiguously be associated with subsequent time-slices.

A basic version of the DPIF as well as a prototype of the service configuration tool have been implemented and are currently being enhanced in the context of the FP7 DIADEM project. In this project we are investigating incorporation of advanced negotiation techniques as well as integration of Multi Criteria Decision Analysis and Scenario Based Reasoning methods facilitating human-based processing in workflows.

7 Acknowledgments

The presented work is partially funded by the European Union under the Information and Communication Technologies (ICT) theme of the 7th Framework Programme for R&D, ref. no: 224318.

References

[1] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[2] Shawn Bowers and Bertram Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In *DILS*, pages 1–16, 2004.

[3] Jorge Cardoso and Amit P. Sheth. Semantic e-workflow composition. *J. Intell. Inf. Syst.*, 21(3):191–225, 2003.

[4] David Chiu and Gagan Agrawal. Enabling ad hoc queries over low-level scientific data sets. In *SSDBM*, pages 218–236, 2009.

[5] David Chiu, Sagar Deshpande, Gagan Agrawal, and Rongxing Li. A dynamic approach toward qos-aware service workflow composition. In *ICWS*, pages 655–662, 2009.

[6] Tina Comes, Claudine Conrado, Michiel Kamermans, Gregor Pavlin, Niek Wijngaards, and Michale Hiete. An intelligent decision support system for decision making under uncertainty in distributed reasoning frameworks. In *7th International Conference on Information Systems for Crisis Response and Management*, Seattle, USA, May 2-5 2010.

[7] Subrata Das. *High-Level Data Fusion*. Artech House, Inc., Norwood, MA, USA, 2008.

[8] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, Kent Blackburn, Albert Lazzarini, Adam Arbree, Richard Cavanaugh, and Scott Koranda. Mapping abstract complex workflows onto grid environments. *J. Grid Comput.*, 1(1):25–39, 2003.

[9] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.

[10] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[11] Nicholas R. Jennings, Katia P. Sycara, and Michael P. Georgeff. Editorial. *Autonomous Agents and Multi-Agent Systems*, 1(1):5, 1998.

[12] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.

[13] Michiel Kamermans. Distributed perception networks: Effecting consistent agent organisation and optimising communication volume in a distributed multi-agent network setting. *Masters Thesis, Informatics Institute, University of Amsterdam*, 2008.

[14] Philip Maechling, Hans Chalupsky, Maureen Dougherty, Ewa Deelman, Yolanda Gil, Sridhar Gullapalli, Vipin Gupta, Carl Kesselman, Jihie Kim, Gaurang Mehta, Brian Mendenhall, Thomas A. Russ, Gurmeet Singh, Marc Spraragen, Garrick Staples, and Karan Vahi. Simplifying construction of complex workflows for non-expert users of the southern california earthquake center community modeling environment. *SIGMOD Record*, 34(3):24–30, 2005.

[15] Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, 2007.

[16] G. Pavlin, P. de Oude, M. Maris, J. Nunnink, and T. Hood. A multi agent systems approach to distributed Bayesian information fusion. *International Journal on Information Fusion*, 2008. To appear.

[17] Gregor Pavlin, Frans C. Groen, Patrick de Oude, and Michiel Kamermans. *A Distributed Approach to Gas Detection and Source Localization Using Heterogeneous Information*. Springer Verlag, 2010. ISBN 978-3-642-11687-2, in print.

[18] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, 1988.

[19] Mihnea Scafeş and Costin Bădică. Preliminary design of an agent-based system for human collaboration in chemical incidents response. In Ulrich Ultes-Nitsche, Daniel Moldt, and Juan C. Augusto, editors, *Proc.of MSVVEIS 2009 – 7th Int.Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*. INSTICC Press, 2009.

[20] Mihnea Scafeş and Costin Bădică. Service negotiation mechanisms in collaborative processes for disaster management. In *Proceedings of the 4th South East European Doctoral Student Conference (DSC 2009), Research Track 2: Information and Communication Technologies*, Thessaloniki, Greece, 2009. SEERC.

[21] Mihnea Scafeş and Costin Bădică. Conceptual framework for design of service negotiation in disaster management applications. In *To appear in Proc. of 2nd International Workshop on Agent Technology for Disaster Management (ATDMâĂŹ09)*, Studies in Computational Intelligence. Springer Verlag, 2010.

[22] Quan Z. Sheng, Boualem Benatallah, Marlon Dumas, and Eileen Oi-Yan Mak. Self-serv: A platform for rapid composition of web services in a peer-to-peer environment. In *VLDB*, pages 1051–1054, 2002.

[23] Ronny Siebes, David Dupplaw, Spyros Kotoulas, Adrian Perreau de Pinninck Bas, Frank van Harmelen, and David Robertson. The openknowledge system: an interaction-centered approach to knowledge sharing. In *Proceedings of the 15th Intl. Conference on Cooperative information systems (CoopIS) in OTM 2007*, 2007.

[24] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113, 1980.

[25] Paolo Traverso and Marco Pistore. Automated composition of semantic web services into executable processes. In *International Semantic Web Conference*, pages 380–394, 2004.

[26] Chris J. van Aart, Bob Wielinga, and Guus Schreiber. Organizational building blocks for design of distributed intelligent system. *International Journal of Human-Computer Studies*, 61(5):567 – 599, 2004.

[27] W.M.P. van der Aalst, A.H.M. ter Hofstede ans B. Kiepuszewski, and A.P. Barros. Workflow patterns. *Distributed and Parallel Databases*, pages 5–51, 2003.

[28] José M. Vidal. *Fundamentals of Multiagent Systems: Using NetLogo Models*. 2006. http://www.multiagent.com.

[29] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.