Interaction Protocols for Cross-organisational Workflows☆

Flávio Soares Corrêa da Silva[a], Mirtha Lina Fernández Venero[a], Diego Mira David[a], Mohammad Saleem[b], Paul W. H. Chung[b]

[a] *University of São Paulo, Brazil 05508090*
[b] *Loughborough University, UK LE11 3TU*

Abstract

Workflow technologies are widely used in industry and commerce to assist in the specification, execution and completion of well defined processes within organisations. As industrial and commercial relations have evolved, based on advances on information and communications technologies, cross-organisational workflow integration has become an important issue. Since organisations can have very different workflows, the creation of compatible workflows so that organisations can collaborate and/or carry out mutual transactions automatically in an integrated fashion can be a very complex and time consuming process. As a consequence, the development of technologies to support the creation and execution of compatible workflows is a most relevant issue. In the present article we introduce the **JamSession** coordination platform as a tool to implement cross-organisational workflow integration. **JamSession** is declarative and based on algebraic specification methods, and therefore workflow integration implemented using this platform can profit from formal behavioural analysis, based on which desired features and properties can be verified and/or obtained.

1. Introduction

Workflow process definitions form an important part of organisational knowledge bases, as they specify how activities are sequenced and who is responsible for carrying out each activity within an organisation. Workflow technologies are widely used in industry and commerce to assist in the specification, execution and completion of well defined processes within organisations (WfMC (1995)). With the advances in communication technologies, electronic business is on the increase and there is a need for efficient and effective workflow tools to support all sorts of business transactions. However, since organisations can have very different workflows, the creation of compatible workflows so that organisations can collaborate and/or carry out mutual transactions automatically in an integrated fashion can be a very complex and time consuming process. As a consequence, the development of technologies to support the creation and

☆This work has been partially funded by FAPESP and Microsoft Research through the **JamSession** project. The authors thank Dr. Shaheen Fatima for comments and suggestions on early versions of the present article.

execution of compatible workflows is a most relevant issue. A company can have many business partners, and the interactions between the company and its partners can be either ad hoc or on a permanent basis. Therefore, it is critical that a flexible approach is used for workflow integration, lest the required overhead to set up the collaboration can overcome the benefit from the corresponding mutual transactions.

The main thrust of this article is on the execution of workflow interactions across organisational boundaries – heretofore referred to as *cross-organisational workflows*. Such interactions should be flexible and reliable, as well as distributed and loosely coupled. Loose coupling ensures that organisations require little or no knowledge about each others' internal activities, and thus a greater freedom for changing their internal processes without affecting the overall collaboration. A distributed approach for the management of interactions ensures that organisational resources remain independent and, therefore, organisations have greater control over their processes. Distributed coordination mechanisms for cross-organisation workflows (e.g. Chen (2008)) have shown to be more effective than centralised ones, since the latter usually entail tighter coupling (van der Aalst (1999); Schulz and Orlowska (2004); Biegus and Branki (2004)).

Recently, a platform has been proposed for the integration and coordination of heterogeneous computational resources (da Silva (2011)). It has been named **JamSession**, after the standard practice of jazz musicians inaugurated around 1940-1950 in New York (USA), who got together in certain specific pubs to form new and innovative band formations after their hired performances. The **JamSession** platform aims at being for software components what those pubs were for the jazz musicians, namely a meeting point where previously existing components can get together to form new and innovative service systems. The main features of **JamSession** are its *(1)* formal and declarative foundations, *(2)* simplicity and *(3)* usability. **JamSession** is based on the construction and execution of Knowledge-based Interaction Protocols (KBIPs), which can be designed using a straightforward and user-friendly graphical language which is also part of the platform. The platform is compact and lightweight so it can be implemented using low computational resources.

In this article, we show how **JamSession** can be used to implement the integration of workflows, thus being an ideal tool for the specification and execution of cross-organisational workflows. The goal is to show a set of KBIPs which can be used to mediate the interaction between organisational workflows, based on which we illustrate the applicability of **JamSession** for cross-organisational workflow integration. The mediation must ensure that the workflows remain decoupled and independent – except, of course, for the synchronous dependency characterised by control flow patterns. Furthermore, we require that a *separation of concerns and responsibilities* is built in the protocols, so that each organisation builds the interaction protocols capable of directly exchanging information with its internal workflow – this way ensuring required trust and security levels.

Hence, the contributions of this article correspond to the introduction of **JamSession** as a tool for cross-organisational workflow integration, with the features of being formally verifiable, easy to use and sufficiently flexible as to provide an environment for the specification and execution of integration patterns which ensure proper separation of concerns and responsibilities across collaborating organisations.

For the sake of completeness, the next section provides an overview of research that has been carried out to address the problem of compatible workflow generation and execution, in the context of cross-organisational workflow integration. An informal but complete description of the **JamSession** platform is provided in Section 3, with illustrative examples of how it works. The central section in this article is section 4, in which we show how cross-organisational workflows can be coordinated through the use of KBIPs. To this end, we consider two of the three main patterns for workflow collaboration described by Chen (Chen (2008)), namely the hierarchical pattern and the peer-to-peer pattern. In Section 5 we sketch out how **JamSession** protocols for cross-organisational workflows can be formally verified, using coloured Petri nets. We conclude the article with some remarks and plans for future work.

2. Workflow Integration

As observed in the previous section, cross-organisational workflow integration is a most relevant – as well as a rather challenging – issue for industry and commerce. In the present section we characterise this field, briefly surveying its origins and recent approaches. We also introduce the advancements in this field which can be explored using the **JamSession** coordination platform.

Workflow management has two main stages, namely build time and runtime (WfMC (1995)):

1. Build time refers to the time period during which workflows are defined and modelled. Build time specifications are modelled using workflow definition tools, which support the specification of workflows using a specification language directly or via a graphical frontend. Workflows built using definition tools and specification languages are executable by workflow management systems (WfMS). WSFL, WfMC's XPDL and BPMI's BPML are some of the most widely used workflow definition tools and corresponding specification languages. Build time specifications are, essentially, a set of routing commands which can be executed by a WfMS.
2. Runtime is the operational stage during which process instances are actually created and managed. The runtime control functions of the WfMS handle the workflow processes in the operational environment and follow the sequential, branching and parallel process coordination patterns specified in the workflow model. The workflow enactment services of the WfMS also ensure the allocation of work items to users and invocation of applications whenever necessary. IBM WebSphere MQ Workflow, FileNET P8 BPM Suite, Staffware Process Suite, TIBCO InConcert and Enhydra Shark are among the most widely used WfMS.

Workflow modelling, revision and update are highly time consuming. In order to improve the efficiency in these activities, the focus of workflow modelling has gradually shifted to process driven services, in order to exploit service oriented architectures and web service architectures (Chen and Yang (2005); Saleem *et al.* (2010); Sirin *et al.* (2003, 2004); Wang *et al.* (2006)).

Service oriented architectures and web service architectures make it possible to automatically generate workflows. Service oriented architectures can be seen as a problem solving paradigm that leverages service based computing standards for delivering business functions and processes as loosely coupled shared services (Krafzig *et al.* (2004)). The web services composition problem can be envisaged as an AI planning problem (Sirin *et al.* (2004)). If each activity in a workflow is represented as an atomic web service, workflow generation problems can be treated as web services composition problems, in which:

1. a planner reasons about a pool of available services,
2. a composition of services that can bring about the desirable state is added into the workflow, and
3. the execution of the workflow achieves the desired goal (Chen and Yang (2005)).

Sirin et al. (Sirin *et al.* (2003)) presented a semi-automatic system for web services composition, which suggests to the user new web services to be included in the workflow. The suggestions are based on constraints specified by the user on attributes of the web services. In more recent work, Sirin et al. (Sirin *et al.* (2004)) extended the semi-automatic system to a fully automatic system. Their system accepts as input high level goals from a user and creates a plan to reach the goals based on the domain description. The system executes the plan using web services extracted from the web. They also presented a sound and complete algorithm to translate OWLS process definitions into SHOP2 domain descriptions. Their proposed system builds plans for one organisation only and does not take multiple organisations into account. They also do not provide support for parallel plans.

Transplan[1] is another system which creates plans from OWLS process definitions to solve high level goals provided by users. Transplan uses SHOP2 for planning. Transplan uses the translation algorithm presented by Sirin et al. (Sirin *et al.* (2004)) to translate OWLS process definitions to SHOP2. It is assumed in Transplan that the preconditions of every operator become false when the operator is added into the plan, which is not always the case in real world domains. Transplan also does not support collaboration between multiple organisations and it is not able to create all possible plans in situations in which parallel plans are possible.

Since organisations have to work in collaboration with each other, the idea of cross organisational workflows arises as a most relevant issue. Most of the existing work on cross organisational workflow collaboration mainly deals with build time collaboration among existing workflows. Earlier research focuses on finding common sequences of activities in existing workflows (van der Aalst (1999); Byde *et al.* (2002); Krukkert (2003)). Existing research also reports coordination which is built through manually discussing the business processes and reaching a business agreement (Schulz and Orlowska (2004)). More recent research has targeted reconciling existing incompatible workflows (Chen and Chung (2007)). The reconciliation, if successful, builds compatible workflows out of the previously incompatible ones. Two or more workflows are said to be compatible if they have an agreed sequence of activities. If workflows are not

[1] See http://sourceforge.net/projects/transplan/.

compatible, then the respective organisations cannot proceed in mutual business (Yang and Papazoglou (2000)).

RosettaNet is a consortium of major electronic companies, which has built the Partner Interface Processes (PIPs) standard. This standard defines a broad set of supply chain processes and data elements[2]. PIPs also defines common interface tasks for supply chain collaboration. Interface tasks refers to activities in which information is exchanged between interacting workflows (Chen (2008)). Workflow collaboration built using PIPs promotes decoupling of workflows, which is a most desirable design feature.

van der Aalst (van der Aalst (1999)) proposed the utilisation of Message Sequence Charts to capture the communication structure in cross organisational workflows. According to van der Aalst, organisations should agree on a common public workflow and organisations can have their independent private workflow within their organisations.

Byde et al. (Byde *et al.* (2002)) proposed a negotiation framework which claims to conduct automatic negotiation over B2B processes. A joint process is created from the interacting processes and the joint process is compared to individual processes to detect any differences and the cost required to remove those differences. This is actually an extension of the approach proposed by van der Aalst (van der Aalst (1999)). The issue with the approaches proposed by van der Aalst and Byde et al. is that both these approaches only target differences caused by different activity content. Any difference regarding activity sequence is considered irreconcilable.

Krukkert (Krukkert (2003)) proposed a solution, in the context of the openXchange project, which takes two activity diagrams as input and compares them to find out all common execution sequences. If any common sequence is found, then a common activity diagram is constructed for collaboration based on the discovered common sequences. This approach assumes all activities to be atomic and represent activity diagrams as state transition systems (Pratt (1991)). The issue with this solution is that it is not able to handle situations in which no common sequence is found.

The works reported by van der Aalst, Byde et al. and Krukkert have a common problem. There must be a common activity sequence in the workflows or activity diagrams of the participating organisations. In a situation such that there is no common sequence, collaboration cannot proceed. This makes the participating workflows or activity diagrams highly coupled with each other. Novel mechanisms for the reconciliation of conflicts are required in order to build decoupled (or, at least, not so strongly coupled) collaboration among workflows.

Schulz and Orlowska (Schulz and Orlowska (2004)) proposed a three tiered cross organisational workflow model, comprised of *coalition workflows*, *workflow views* and *private workflows*. Coalition workflows are constructed on the basis of agreements among the business partners and are made up of abstract services. Individual partners can choose tasks from the coalition workflows which they decide to implement privately. Based on a coalition workflow, relationships between the chosen tasks are obtained. Splits and joins are added to workflow views, which are published to be viewed by the business partners. Workflow views and private workflows are then con-

[2]See RosettaNet Business Dictionary, RNBD v2.1, 2002.

nected through state dependencies. This model requires interacting partners to meet and decide on business agreements before the workflow collaboration can be implemented. This can be a highly time consuming process, especially when the network of business partners grows.

Chen and Chung (Chen and Chung (2006)) presented a framework for collaboration among existing workflows of multiple organisations in order to make them compatible. The approach is based on reconciling existing workflows to bring about compatibility and supporting runtime execution of the compatible cross organisational workflows. A collaboration software agent initializes an interface process offer to a candidate provider, who then evaluates the offer and creates a counteroffer. The requester either accepts or rejects the offer. The process of offer generation, counter-offer generation, acceptance and rejection goes on recursively until negotiation is terminated or reconciliation is achieved. If compatibility is attained, partners move on to fulfilment stage by enacting their workflows. This framework automates the collaboration process and saves time and resources for organisations. This framework helps in decoupling the interacting organisations from each other by automatically building collaboration among their initially incompatible workflows. However, the organisations are still involved in accepting or rejecting offers and counter-offers every time process reconciliation is carried out. Moreover, organisations must model their workflows prior to collaboration, which is a time consuming task.

Wang et al. (Wang *et al.* (2006)) proposed a system for inter-organisational workflow coordination and dynamic composition of workflows. They have introduced dynamic flexibility to the workflow runtime execution stage. Initially, dynamic flexibility was only targeted at the design stage. They used intelligent agents for the dynamic composition of workflows and negotiation of web services over the Internet. Intelligent agents are autonomous problem solving entities which take the state of their environment as input and act on the environment to fulfil certain roles (Jennings (2001)). Wang et al. used intelligent agents to discover, execute and monitor web services. They targeted inter organisational coordination from the perspective of a single organisation which deals in a heterogeneous environment with ad hoc external processes. They did not take the creation of compatible workflows for multiple organisations into account.

Jiang et al. (Jiang *et al.* (2010)) used process-views for cross organisational workflow management. Process-views are abstract processes used to encapsulate sensitive private details of the workflows and expose the details necessary for collaboration. Jiang et al. proposed an approach based on the integration of process-views and timed colored petri-nets for workflow management. They consider both the control and data aspects of the cross organisational workflow management. They simplify the complex workflows with colored petri-net refinement, model the petri-net workflow models to process-view workflow models and then execute the workflow instances. Their approach keeps the autonomy and the privacy of the internal workflows to the organisations while realising the interoperability through the interfaces of the workflow views.

In their earlier work, Jiang et al. (Jiang *et al.* (2008, 2009)) applied the integrated approach of process-views and petri-nets for collaborative product management. Similarly, Chebbi et al. (Chebbi *et al.* (2006)) used workflow views for inter-organizational workflow cooperation. In the same way, the Cross-Work project adopted a view based approach for dynamic business network process management in instant virtual enter-

prises (Eshuis and Grefen (2008); Grefen *et al.* (2009)).

Recently, some work has been done on composing web services into workflows for multiple collaborating organisations. Chen et al. (Chen *et al.* (2011)) suggested a Pi-Calculus based approach to compose web services into cross organisational business processes. A cross organisational business process is modelled as a set of concurrent local processes, which has a global start and a global end activity. The activities in the local processes can receive external start messages. A cross organisational controller controls the flow of control and data in the cross organisational process. The basic problem with this approach is that it is a static modelling approach and the web services composition is not automatic.

Saleem et al. (Saleem *et al.* (2010)) developed a framework that uses intelligent agents to create compatible workflows for multiple collaborating organisations. Their developed framework uses SHOP2 for planning. An instance of intelligent agent is created for every organisation, which works on behalf of the respective organisation. Any step that makes the workflow incompatible with the workflows of the collaborating organisations is discarded by the agents, and a new step is tried. The problem with this framework is that it is not known whether this is a feasible and practical approach, since the coordination among agents before adding every step will lead to very intensive communication among the agents.

Saleem et al. ((Saleem *et al.* (2011); Saleem (2012)) presented a framework that is able to generate compatible workflows for multiple collaborating organisations, from their process definitions and high level organisational goals. The framework also supports collaborative runtime enactment of the generated workflows. The framework relieves organisations from modelling and reconciling workflows each time they interact with new organisations. From the given goals, the framework generates multiple sets of accurate, valid and compatible workflows for the collaborating organisations and gives the users the choice to select the best suitable compatible set of workflows for their business scenario. The framework uses AI planning and web services architecture for the generation of compatible workflows. The framework uses an extended version of SHOP2 planner for planning.

Since workflows need to be enacted, only build time coordination is not enough. There needs to be runtime coordination among interacting organisations, so that the transfer of files and information can happen smoothly and the sequential, parallel or branching navigation of cross organisational activities can be followed. One way to achieve this is to treat business partners as workflow participants and expand the standalone centralised enactment service across the organisational boundaries, as discussed by Chen (Chen (2008)). In this situation, business partners need to share private data, common process definition and a centralised workflow engine. This approach couples the business partners very closely. It is also a very expensive and rigid approach. Sub-flow invocation mechanisms are another alternative in which a chained process is started and all the nested sub-flows in the hierarchical workflow are completed. This approach is technically sound but limited.

Chen and Hsu (Chen and Hsu (2001)) proposed a collaborative process manager which is based on a workflow case transfer approach. In this manager, it is assumed that all interacting partners must have the same definition of workflow, and each organisation is responsible for executing its own activities through its local workflow en-

gine. The organisations recognise their activities on the basis of role matching. Once the organisations finish their activities, they inform the interacting organisations. The communication between partners happens through messages. The problem with this approach is that it assumes homogenous workflow engines among all interacting organisations. This approach also needs the mainstream specification languages to be extended in order to include a collaborative process definition language.

Chen and Chung (Chen and Chung (2006)) proposed a bottom-up approach for cross-organisational workflow enactment. Their approach is WfMS independent and workflow enactment is done via progressive linking, which is enabled by run-time agents. Each interaction point is modelled as an interface activity and agents make sure that outgoing data and incoming data are delivered to the corresponding activities accordingly. A form filling approach is used to ensure this. The corresponding agents fill in information such as activity ID, interaction identifier and general data, and attach any relevant documents to the form after the activity is executed. The form represents the progress of interoperation and can be used for historical record. Chen and Chung assumed that the cross organisational workflows to be enacted were already compatible and the issue of compatibility is solved in build time.

As discussed, existing research has targeted the area of workflow integration extensively. There are a few issues that still remain neglected.

1. While the researchers have focussed on workflow integration to bring about business collaboration, formal verification and analysis of the integration process has not been targeted. Formal verification and analysis ensures the correctness and validity of the workflow integration, which is beneficial in the case of huge coordinating workflows where the correctness and validity cannot be guessed intuitively. Formal foundations ensure the selection of the most efficient integration pattern. JamSession targets formal verification and analysis, due to its formal and declarative foundations it is highly efficient and lightweight.
2. The existing systems are mostly tightly coupled (van der Aalst (1999); Schulz and Orlowska (2004); Biegus and Branki (2004)). JamSession decouples the workflows by only allowing the respective organisation-owned knowledge protocol to have access to the internal details of its workflow. The integration will not be affected even if the internal workflow is changed, as long as the connectors remain unchanged.
3. Most of the runtime integration systems require the coordinating workflows to use homogenous workflow management systems. JamSession integrates heterogeneous workflows.
4. One of the very powerful features of workflow integration is control over the integration process. For example an organisation might be collaborating with multiple business partners at the same time. This means that multiple instances of the workflow of the organisation will be in collaboration with the instances of the workflows of the partner organisations concurrently. Such control over workflow integration has largely been ignored in literature. JamSession enables such powerful control over integration.
5. Most of the existing integration systems support workflow integration either at build time or runtime (Saleem (2012)). The separation of build time and runtime

stages costs time and resources to the interacting organisations (Saleem (2012)). JamSession is one of the very few systems that are able to support workflow integration at both build time and runtime stages. JamSession combines efficiency, de-coupling and usability into one system.

The **JamSession** platform adds to the initiatives above – especially the one proposed by Chen and Chung – by providing a concrete, expressive and user friendly platform to build executable specifications of runtime cross-organisational workflows. Our interest and approach are to provide systems engineers with a concrete tool to enable the high-quality design and implementation of workflow integration patterns, which require minimal computational resources and are formally verifiable with respect to their requirements.

In the next section we describe in detail the **JamSession** platform.

3. The JamSession Platform

JamSession is a platform for the integration and coordination of computational resources. It was conceived initially for highly interactive systems – such as multiplayer videogames – and later extended to manage generic computational resources. The main features of **JamSession** are:

- *High computational performance*, to provide the means for the coordination and integration of software components in realtime systems.
- *Usability*, to ensure that novice users are capable of integrating and coordinating heterogeneous resources with minimised effort.
- *Formal and declarative foundations*, to provide the means for formal design, analysis and verification of coordination protocols.
- *Decoupled integration* of heterogeneous computational resources, to ensure their independence and support a separation of concerns and responsibilities when designing and implementing workflow integration.

Each of these features has been accounted for by previous initiatives. The **JamSession** platform advances upon previous work on workflow coordination and integration, by presenting all features in a single platform, which is also computationally lightweight.

The development of **JamSession** as a general purpose coordination platform has been inspired by specific existing research initiatives, which have accounted for specific subsets of these features, as detailed in the following paragraphs.

A conceptual architecture that bears several similarities with our initiative is that of Electronic Institutions (Esteva *et al.* (2001)). Electronic Institutions are constituted by agents, roles, normative rules that characterise the institutions and scenes in which regulated dialogues occur involving agents. For an agent to participate in a dialogue, it must assume a specific role that is compatible with that dialogue, and in order to assume a specific role the agent must ensure that it has the necessary capabilities required for that role. Once all roles required for a dialogue are fulfilled by agents, the dialogue can

start and, if all norms are followed, the dialogue can run to its end, thus performing a structured and controlled sequence of linguistic actions.

In **JamSession**, the concept of role is replaced by the notion of location, as described in the paragraphs below. Agents have to move to specific locations in order to participate in collaborative actions, as some of their capabilities become available only in those locations.

An additional relation between Electronic Institutions and **JamSession** is that both have tackled explicitly usability issues. Electronic Institutions have been used as the foundation to build user friendly systems, based on user interfaces that resort to interactive animations as means to bring to users the experience of interacting with services as if interacting with human clerks in a simulated virtual environment (Bogdanovich *et al.* (2007)).

Norm-mediated interactions have been explored in a variety of initiatives (see e.g. (da Silva and Vasconcelos (2006); Vasconcelos *et al.* (2009))), in which interaction protocols are grounded by normative rules that specify permissions, obligations and prohibitions to which agents must conform. In **JamSession**, this concept is simulated as interaction protocols attached to specified locations.

The Lightweight Coordination Calculus (LCC) (Robertson (2004)) is a language to build executable algebraic specifications for interactions among agents in multiagent systems. It is based on formalisms such as the Ambient Calculus and the π-Calculus, and it has been proved to have similar expressive power to Electronic Institutions. It has been used extensively in a variety of applications, such as bioinformatics and emergency response.

The main difference between the LCC and **JamSession** is that the former does not account explicitly and directly for the notion of mobility and situatedness. LCC has lately been extended to account for situatedness explicitly, thus resulting in a language coined Ambient LCC, which however seems not to have been much used in practice. This could be because it has become a little unfriendly and difficult to implement.

The Multilayered Multiagent Situated Systems architecture (MMASS) (Vizzari (2004)) considers mobility as the central notion for situatedness and interactions in multiagent systems. In MMASS we find the notion of locations, which are structured by pathways forming a graph of sites through which agents can move to look for specific resources to accomplish their goals. We have borrowed these concepts for **JamSession**.

A fundamental notion in **JamSession** is the concept of *location*. Intuitively, we have agents in an environment, whose capabilities are blocked or released for use, depending on the locations where they are situated. When a user needs a specific service, they must make sure that the agent that has the capability of furnishing that service in a specific location has actually moved to the appropriate location. Locations are, therefore, abstractions of groups of services, whose accessibility is controlled by how users transport named agents to/from specified locations.

Formally, we have a directed graph to specify locations and their connections. The nodes of the graph are the locations, and the arrows characterize the admissible transitions that agents can perform to move across locations. **JamSession** is a coordinator of resources, which are represented as capabilities of situated agents. An agent stays in a location until it receives an order to move to a different location.

We employ, in the presentation of **JamSession**, the PROLOG convention for terms and variables. Hence, terms starting with capital letters are free variables. An order for an agent to move is a triple as follows:

$$move(Agent, Location_1, Location_2)$$

In the order above, the agent $Agent$ is assumed to be in $Location_1$ and is being requested to move to $Location_2$. An order to move can be evaluated, in which case an attempt to execute it shall be performed and a corresponding truth value shall be assigned to it, depending on the success of the execution. If the agent $Agent$ is indeed in $Location_1$ and there is a direct link from $Location_1$ to $Location_2$, then $Agent$ is moved from $Location_1$ to $Location_2$, and the order is evaluated to $\top$. Otherwise, $Agent$ stays wherever it is and the order is evaluated to $\bot$.

The capabilities of situated agents are represented as first-order *predicates* in **JamSession**. Each predicate is associated to a pair $[Agent, Location]$. Predicates also have *Input* and *Output* parameters, which are formed respectively as first-order terms and free variables. A predicate is as follows:

$$[Ag, Loc]predicate((I_1, ..., I_m), (O_1, ..., O_n))$$

In this predicate, Ag is an agent, Loc is a location, I_i are input terms and O_j are output variables.

Predicates are defined during the design of a system, specifying what resources can be triggered by what agents and in which locations. During the execution of a system, predicates are used to actually activate resources.

A predicate can be triggered, i.e. there can be an attempt to evaluate it, at any time. Most typically, predicates are used as attempts to activate system resources. The predicate input terms are syntactically and semantically verified, and then it is checked whether the agent Ag is located in Loc and if Ag has the permission to activate the predicate in Loc. If all verifications are successful, then the corresponding resource is activated, possibly instantiating the output variables, and the predicate is evaluated either to $\top$ or to $\bot$, depending on its programmed behaviour. If the terms are not sound, or if Ag is not located in Loc, or if Ag does not have the permission to trigger the predicate in Loc, then the predicate is blocked and the corresponding resource cannot be activated. In this case, the predicate is evaluated to $\bot$ and the output variables are returned uninstantiated[3].

Predicates and movements are combined in **JamSession** using *knowledge-based interaction protocols* (KBIPs). A KBIP is a structure of entities that specifies their order of evaluation. Entities can be of three types:

1. Orders for agents to move.
2. Predicates.
3. Recursively, other KBIPs.

[3]In the full-fledged specification of **JamSession**, special predicates are included to enable asynchronous communication between agents. Since this feature is not used in the present work, we are not going to detail the asynchronous communication predicates in the present article.

KBIPs are linked to locations. A request to trigger a KBIP can result in the following alternative situations:

- The requested KBIP is not actually defined for the specified location. In this case, the obtained truth value is $\bot$.
- The requested KBIP is defined for the specified location. In this case, the specification of the KBIP is retrieved and evaluated, based on the algebraic rules that govern the behaviour of the connectives that are used in the specification of the KBIP. The result of the evaluation determines the truth value that shall be assigned to the KBIP, which can be $\top$ or $\bot$.

A KBIP is denoted as follows:

$$[Loc]kbip((I_1, ..., I_m), (O_1, ..., O_n))$$

Here, Loc stands for the location to which the KBIP is connected, I_i are first order input terms and O_j are output variables as before. The expected utilization of terms and variables in the specification of KBIPs is for parameter passing across predicates and KBIPs.

A KBIP takes the form of a formula in disjunctive normal form, in which atoms are the entities in the list above (namely, orders for agents to move, predicates and KBIPs). More formally, the specification of a KBIP takes the following form:

- $[Loc]kbip((I_1, ..., I_m), (O_1, ..., O_n)) ::= \bigvee_1^k F_i$.
- $F_i ::= \bigwedge_1^{r_i} e_j$.
- $e_j ::= Move|Predicate|Kbip$.
- $Move ::= move(Agent, Location_1, Location_2)$
- $Predicate ::= [Ag, Loc]predicate((I_1, ..., I_n), (O_1, ..., O_n))$
- $Kbip ::= [Loc]kbip((I_1, ..., I_m), (O_1, ..., O_n))$

Both disjunction and conjunction are assumed to be non-commutative, therefore the order in which entities are presented in KBIPs can change their evaluation.

In order to simplify the specification of KBIPs, a graphical language has been devised. Conjunction is drawn as a line connecting two entities, and disjunction is drawn as a black circle. The three basic entities that comprise a KBIP (move, predicate and kbip) are characterised as shown in Figure 1.

For example, the following simple knowledge-based interaction protocol:

$$\begin{aligned}&[loc_1]kbip_1((inp), (X)) ::= \\ &\quad [a, loc_1]pred_1((inp), (X)) \vee \\ &\quad (move(a, loc_1, loc_2) \wedge [a, loc_2]pred_2((inp), (X))\end{aligned}$$

can be represented diagrammatically as shown in Figure 2.

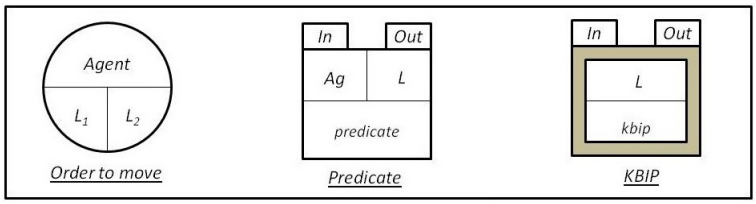

Figure 1: Graphical representations for the three basic entities of a knowledge-based interaction protocol.

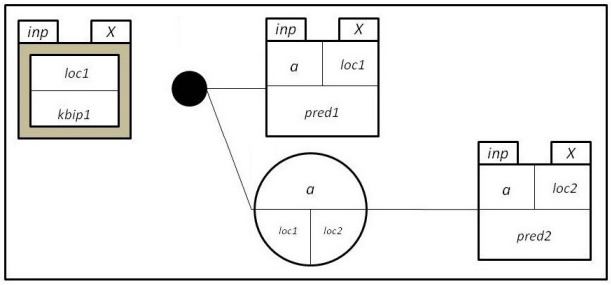

Figure 2: Graphical representation of a simple knowledge-based interaction protocol.

This specification must be complemented by information about what resources belong to what locations and agents, about the links that connect locations and about the initial setting of agents. For example, we could have:

- $[a, loc_1]pred_1((inp), (loc_1))$ ::= (specification of the behaviour associated to $pred_1$).
- $[a, loc_2]pred_2((inp), (loc_1))$ ::= (specification of the behaviour associated to $pred_2$).
- Structure of locations and initial location of agent a as shown in Figure 3.

KBIPs are used to specify, implement and execute interactions among users of information systems based on the integration and coordination of resources. Interaction protocols are also employed to specify, implement and execute interactions between

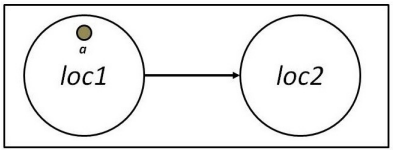

Figure 3: Initial setting for a simple knowledge-based interaction protocol.

users and the information systems themselves. Agents, locations and all entities that comprise the KBIPs are the conceptual resources used to characterise the desired interactions both at design time and run time. KBIPs behave as mediators between computational resources, and ensure that heterogeneous resources are decoupled.

Interaction protocols can be triggered concurrently. This means that more than one protocol – and more than one instance of a protocol – may be running at any time. It shall be left to the system designer to ensure that the resulting concurrent groups of protocols generate behaviours that are in accordance with the requirements specified for each application. The declarative characterisation of knowledge-based interaction protocols in **JamSession**, nevertheless, provides the means for formal analysis and verification of protocols.

We have developed a cloud-based implementation of **JamSession**. In this implementation, a cloud-based server stores and manages the information about the graph of locations, agents, as well as clients and their corresponding KBIPs. Whenever a client subscribes to the server, it exports to the server a collection of interfaces for its locally stored protocols. Whenever a client needs to trigger a protocol, it sends a request to the server, which then verifies the validity of the request. If the request is valid, the server makes use of the corresponding protocol interface and sends a request to the appropriate client – i.e. the one which hosts the required protocol – in order to effectively trigger it. Clients also host the implementation of predicates, whose evaluations are requested by the server in the course of executing protocols. Side effects of the evaluations of predicates – such as file updates and interaction with external computational resources (e.g. sensors and actuators) are also implemented locally in clients.

An overview of **JamSession**'s implementation is depicted in Figure 4. The communication between clients and the server is performed by means of web services, which have been implemented using *Windows Communication Foundation* and *NetTcpBinding*. An implementation of the *TupleSpaces* concept (the *SQLSpaces*[4]) is used for managing the information stored at the server. The whole platform's implementation is based on the *F#* functional programming language and the server can be hosted on the Microsoft Azure cloud platform. Predicates can be specified either as *F#* or *C#* classes. They are compiled independently to executable libraries (DLLs).

4. Interaction Protocols for Workflow Integration

In order to show **JamSession** at work, and illustrate how it can be used to coordinate and integrate cross-organisational workflows, we consider two specific workflow integration patterns, and show how they can be implemented using knowledge-based interaction protocols as mediators. We also show how properties in **JamSession** protocols can be formally verified, through the first (and simplest) workflow integration pattern that we consider. Further details about formal analysis of interaction protocols are presented in the next sections.

We consider the *hierarchical* and the *peer-to-peer* cross-organisational workflow integration patterns (Chen (2008)). The mediation between workflows following the

[4]See `http://sqlspaces.collide.info/`

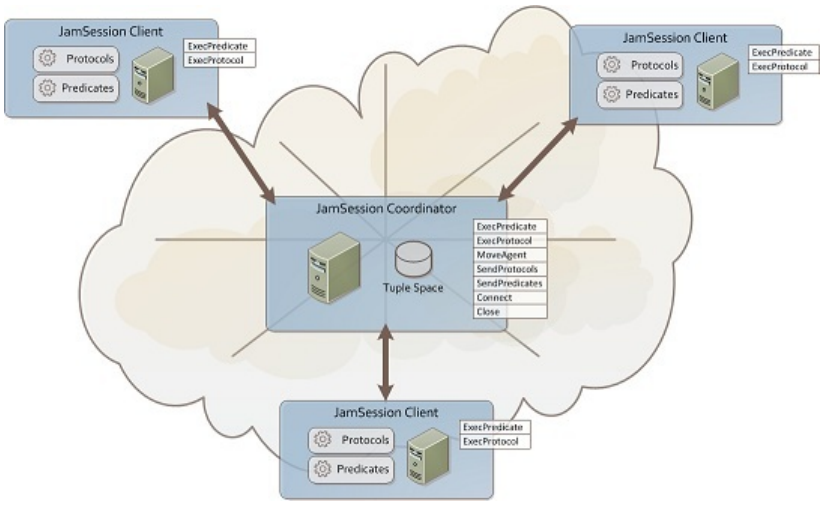

Figure 4: JamSession architecture.

hierarchical pattern is relatively simple to implement and to check with respect to desirable properties. The mediation between workflows whose interaction occurs in a peer-to-peer basis is slightly more sophisticated, as integration must occur between specific *instances* of each workflow, instead of between the workflows as abstract entities.

4.1. The hierarchical pattern

In the hierarchical pattern, an activity belonging to a workflow A triggers a full instance of a workflow B, and waits for the completion of B to release the continuation of A (Figure 5).

Although the pattern is simple, it has many applications. Consider a company that assists its clients in visa application. It would have a process that includes tasks for helping its clients completing the form, gathering the relevant supporting documents, etc. After the preparation is complete, the company would submit the application to the appropriate authority which will then process the application. While the application is being processed, the company's workflow instance for that particular application will be suspended until the authority's workflow instance for processing that particular application is completed and then control is passed back to the company.

Our goal is to build a set of knowledge-based interaction protocols to mediate the interaction between workflow A and workflow B. The mediation must ensure that the workflows remain decoupled and independent – except, of course, for the synchronous dependency characterised by this pattern. Furthermore, we require that a *separation of concerns and responsibilities* is built in the protocols, so that each organisation builds the interaction protocols capable of directly exchanging information with its internal workflow – this way ensuring the required trust and security levels are achieved.

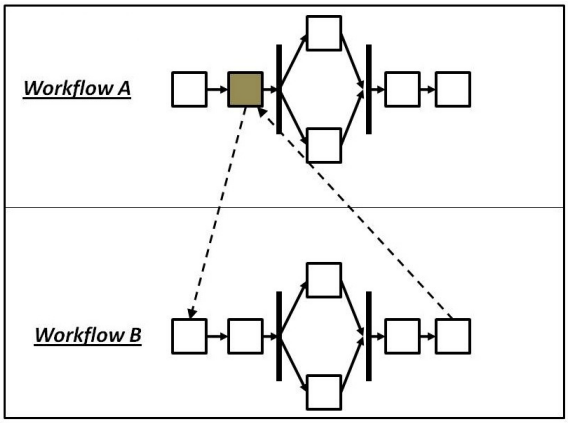

Figure 5: Hierarchical cross-organisational workflow integration pattern.

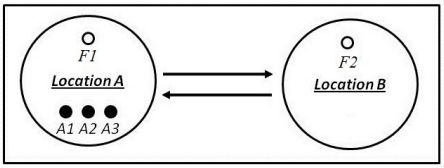

Figure 6: Locations and initial setting for the hierarchical integration pattern.

This pattern can be implemented using three knowledge-based interaction protocols, as follows. As an additional feature, and so that we can demonstrate some additional control that can be built over the mediation between the workflows using **JamSession**, we limit the number of concurrent instances of workflow B that can be triggered. In our concrete example, we limit this number to three, i.e. no more than three instances of B can be triggered simultaneously.

We employ two locations and five agents, initially configured as displayed in Figure 6, in which white dots represent agents that remain fixed in each location, and black dots represent agents that move across locations. There is a correspondence between these locations and the lanes in the workflow (and, therefore, between the locations and the organisations to be coordinated). Since we want to limit the concurrent instances of B to at most three, we have three mobile agents initially in location A. Each time an instance of workflow B is started, one of these agents is moved to location B, and when workflow B is completed the agent is sent back to location A.

The three knowledge-based interaction protocols are presented, respectively, in Figures 7, 8 and 9. The first interaction protocol is owned by the organisation who owns workflow A. It is triggered by the shaded activity in workflow A, and for this reason we name it P_1^A. The activity in workflow A passes a set of values to P_1^A, which employs an internal predicate to verify its consistency. If the input values are consistent then the

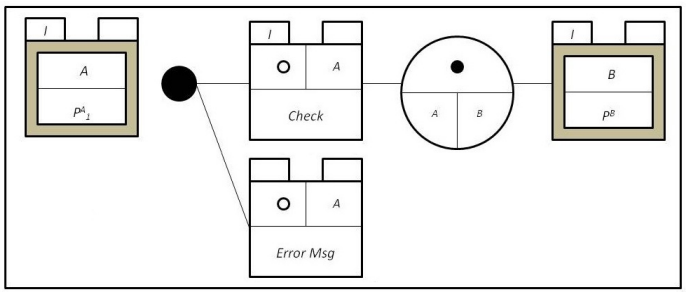

Figure 7: The knowledge-based interaction protocol P_1^A.

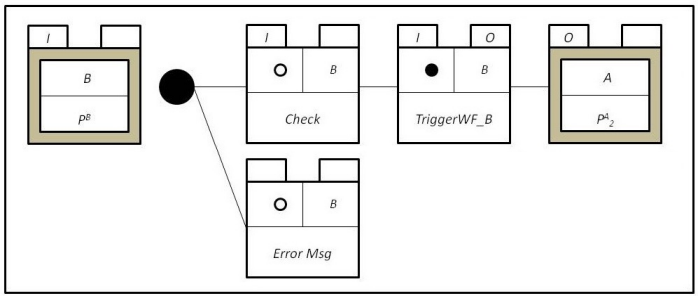

Figure 8: The knowledge-based interaction protocol P^B.

protocol attempts to move one of the mobile agents from location 1 to location 2 – this way book-keeping the number of instances of workflow B that are active. If both previous entities are evaluated as $\top$ then the protocol P_1^A triggers the second interaction protocol that is used to implement the integration pattern.

The second interaction protocol is owned by the organisation who owns workflow B. For this reason, we name it P^B. It is triggered by P_1^A, and it receives from P_1^A the same input values received by that protocol. Since P^B is decoupled and independent from P_1^A, it performs its own consistency check on the set of input values. If the values are consistent, then it triggers a predicate which, in turn, triggers the whole workflow B. Finally, upon successful completion of the workflow – and therefore successful evaluation of the corresponding predicate – it triggers the third and last interaction protocol that is used to implement the integration pattern. We assume that the execution of workflow B generates a set of output values, which are captured by the predicate which triggers it. These values are passed as input parameters to the third knowledge-based interaction protocol.

The third interaction protocol is owned by the organisation who owns workflow A. For this reason, we name it P_2^A. It is triggered by P^B, and it receives from P^B the output of workflow B. It performs its consistency check on the received values and, if the values are consistent, it moves one mobile agent back from location B to location

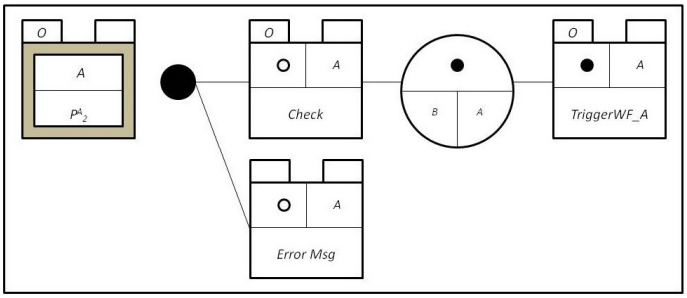

Figure 9: The knowledge-based interaction protocol P_2^A.

A – this way releasing an instance of workflow B for further use. If the movement of the agent from location B to location A is successful then it sends forward the output of workflow B to the shaded activity in workflow A, this way completing the coordination of both workflows.

JamSession interaction protocols are declarative, which simplifies the formal analysis of their behaviour. For example, we can verify that the set of protocols presented above always terminates successfully, i.e. with truth-value $\top$.

The verification goes as follows: the evaluation of predicates and orders for agents to move depends on the location of the mobile agents. Since we have the possibility of concurrent execution of knowledge-based interaction protocols, we must consider, for each protocol, all possible configurations of agents in locations. In our example, we must consider the following configurations:

1. All three mobile agents in location A.
2. Two mobile agents in location A and one agent in location B.
3. One mobile agent in location A and two agents in location B.
4. All three mobile agents in location B.

We examine now the possible outcomes of protocol P_2^A:

1. All three mobile agents in location A:

   The values O received by predicate *Check* are verified.
   If verification *fails*, Then
   The predicate *Error Msg* is triggered.
   This predicate always succeeds with $\top$.
   Else
   The order for an agent to move from B to A is triggered.
   Since there are no mobile agents in B, this order fails.
   The predicate *Error Msg* is triggered.
   This predicate always succeeds with $\top$.
   Hence, the protocol always succeeds with $\top$.

2. Two mobile agents in location A and one agent in location B:

   The values O received by predicate *Check* are verified.
   If verification *fails*, Then
   The predicate *Error Msg* is triggered.
   This predicate always succeeds with $\top$.
   Else
   The order for an agent to move from B to A is triggered.
   This order succeeds, thus changing the configuration to
   All three mobile agents in location A.
   The predicate *TriggerWF_A* is triggered,
   attempting to return the values from workflow B
   to workflow A.
   If the predicate succeeds, then the protocol succeeds with $\top$.
   Else the predicate *Error Msg* is triggered,
   succeeding with $\top$.
   Hence, the protocol always succeeds with $\top$.

3. One mobile agent in location A and two agents in location B:

   The values O received by predicate *Check* are verified.
   If verification *fails*, Then
   The predicate *Error Msg* is triggered.
   This predicate always succeeds with $\top$.
   Else
   The order for an agent to move from B to A is triggered.
   This order succeeds, thus changing the configuration to
   Two mobile agents in location A and one agent in location B.
   The predicate *TriggerWF_A* is triggered,
   attempting to return the values from workflow B
   to workflow A.
   If the predicate succeeds, then the protocol succeeds with $\top$.
   Else the predicate *Error Msg* is triggered,
   succeeding with $\top$.
   Hence, the protocol always succeeds with $\top$.

4. All three agents in location B:

   The values O received by predicate *Check* are verified.
   If verification *fails*, Then
   The predicate *Error Msg* is triggered.
   This predicate always succeeds with $\top$.
   Else
   The order for an agent to move from B to A is triggered.
   This order succeeds, thus changing the configuration to

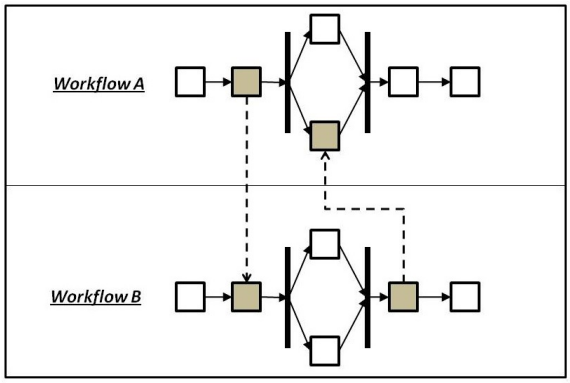

Figure 10: Peer-to-peer cross-organisational workflow integration pattern.

One mobile agent in location A and two agents in location B.
The predicate *TriggerWF_A* is triggered,
attempting to return the values from workflow B
to workflow A.
If the predicate succeeds, then the protocol succeeds with $\top$.
Else the predicate *Error Msg* is triggered,
succeeding with $\top$.
Hence, the protocol always succeeds with $\top$.

This way, all possible outcomes of the knowledge-based interaction protocol P_2^A are verified. Similar analyses can be developed for the other two protocols, thus completing the formal verification that they always complete with truth-value $\top$.

Other properties could be verified. For example, we could be interested in checking whether a certain configuration can be reached, or whether an error message can be generated given specific input values. Verifications of this nature can be developed on **JamSession** protocols, based on configuration analysis as shown above.

4.2. The peer-to-peer pattern

The peer-to-peer cross-organisational workflow integration pattern implements interactions between *instances* of two workflows. In the peer-to-peer pattern, during the execution of a specific instance of a workflow A, an activity belonging to this workflow communicates with an activity belonging to an existing and running instance of a workflow B, this way completing the required input for that activity to be executed and for that instance of workflow B to continue its execution. Both instances of workflows A and B continue their execution concurrently, until an activity in the instance of A requests as input some results from an activity in the instance of B which has previously received a message from A. This way, the running instance of B must further communicate with the instance of A which has provided information to it, thus characterising cooperation between particular instances of two workflows (Figure 10).

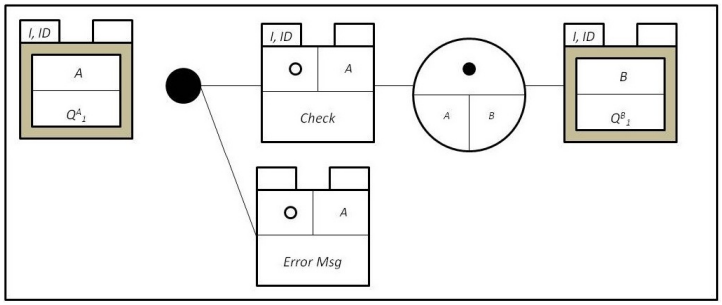

Figure 11: The knowledge-based interaction protocol Q_1^A.

An example of this pattern is the interactions between vendors and customers which is commonly found in manufacturing. The workflow instance of the customer and the workflow instance of the vendor will have a number of interaction tasks. An example of such a task is that the customer sends an acceptance note to the vendor after receiving a quotation. However, the customer does not necessary have to wait for the vendor to deliver the goods before moving to the next task as it could, for example, progress to making a deposit while the vendor is manufacturing the specific goods for the customer. Therefore, the two workflow instances will have tasks that are executed in parallel rather than control flow had to be passed strictly from one workflow to another.

In order to implement this integration pattern, we need to provide an instance identifier for workflow A along with its message to workflow B, so that the instance of B that receives this message can respond to the original instance of A when replying to it. As in the previous example, and just to exhibit additional control over the integration of workflows that can be built using **JamSession**, we limit the number of instances of A and B which can cooperate simultaneously to three.

This pattern can be implemented using four knowledge-based interaction protocols, as follows. The locations, agents and initial setting are exactly as in the previous example (Figure 6).

Similar to the previous example, the first interaction protocol is owned by the organisation who owns workflow A. It is triggered by the shaded activity in workflow A, and for this reason we name it Q_1^A. The activity in workflow A passes a set of values to Q_1^A, which employs an internal predicate to verify its consistency. It is important to highlight that, among these values, workflow A sends through its own ID, which is used in the future to find the appropriate instance of A with which workflow B must communicate.

As before, if the input values are consistent then the protocol attempts to move one of the mobile agents from location A to location B – this way book-keeping the number of instances of workflow B that are active. If both previous entities are evaluated as $\top$ then the protocol Q_1^A triggers the second interaction protocol that is used to implement the integration pattern. This protocol also receives the ID of A. It is presented in Figure 11.

The second interaction protocol is owned by the organisation who owns workflow

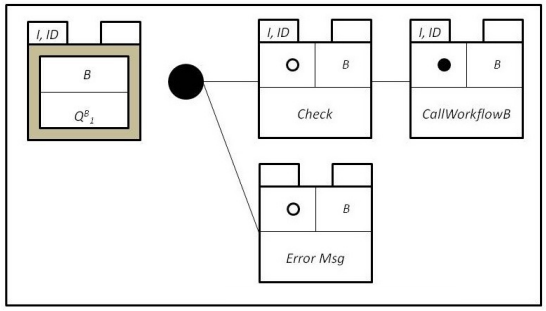

Figure 12: The knowledge-based interaction protocol Q_1^B.

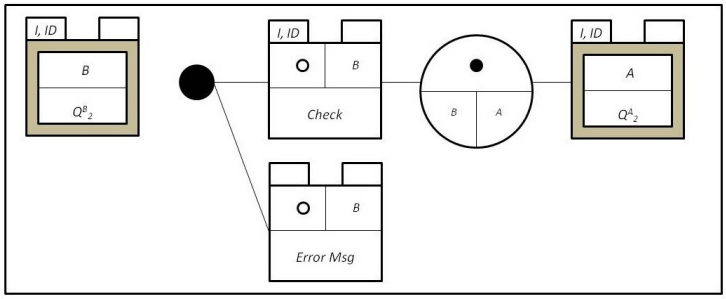

Figure 13: The knowledge-based interaction protocol Q_2^B.

B. For this reason, we name it Q_1^B. It is triggered by Q_1^A, and then checks the consistency of the received input values. If the consistency verification succeeds then it triggers a predicate which sends the input values to the appropriate activity in workflow B. It should be highlighted that the ID of A is also sent to workflow B. We assume that this value is properly stored within workflow B, to be used when a communication from B to A is enacted. This protocol is presented in Figure 12.

The third interaction protocol starts to walk the flow the way back from B to A. It is owned by the organisation who owns workflow B, and is triggered by the second shaded activity in workflow B. We name it Q_2^B. This is the first protocol that uses the ID of A, which is received as an input. As the previous protocols, it verifies the consistency of the input values. If the verification succeeds then it attempts to move back one agent from location B to location A – this way releasing an instance of B for future use. If the movement of the agent from location 2 to location 1 succeeds, then it triggers the fourth and final protocol that implements this integration pattern. The protocol Q_2^B is presented in Figure 13.

The final protocol that implements the peer-to-peer cross-organisational workflow pattern is owned by the organisation who owns workflow A, and is triggered by Q_2^B. We name it Q_2^A. It receives from Q_2^B input values which include the ID of the orig-

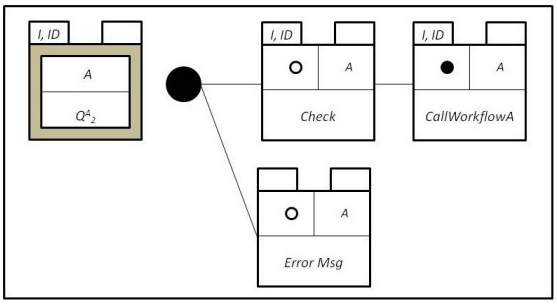

Figure 14: The knowledge-based interaction protocol Q_2^A.

inal instance of workflow A which was initially used to trigger Q_1^A. It verifies the consistency of the received input values and, if the verification succeeds, it triggers a predicate which communicates with workflow A and sends the appropriate input values to it. This final predicate is a little more sophisticated to implement than the previous ones, as it must contain an internal mechanism to search for the appropriate instance of A. The search is performed using the ID of A. Once the appropriate instance of A is found, the interaction protocol sends the appropriate input values to the second shaded activity in A, and releases that protocol so that it can continue its execution. The protocol Q_2^A is presented in Figure 14.

4.3. Example of JamSession Integration with Bonita Open Solution

As referred to in the previous section, a prototypical implementation in *F#* has been developed, based on which several small knowledge-based interaction protocols have been implemented, exhibiting good computational performance. In particular, the above protocols were tested in connection with Bonita workflow definitions[5]. The graphical front end of the platform was used to design the KBIPs (see Figure 15).

The integration with Bonita was implemented based on Java class connectors attached to activities in the workflow definition. For example, for the hierarchical pattern of Figure 5 two connectors were used: one for the activity in the workflow A which invokes the workflow B and the other for the last activity in the worflow B (see Figure 16). These connectors use a **JamSession** auxiliary program to trigger protocols or predicates. For example, in Figure 17, the connector of the $Step2$ in workflow A is used to trigger the KBIP P_1^A. In the reverse direction, i.e. to trigger a Bonita workflow instance from a **JamSession** protocol, the implementations of the *TriggerWF* and *CallWorkflow* predicates use **Bonita Rest API** interfaces such as *runtimeAPI* and *queryRuntimeAPI*.

In the examples presented in this article we have illustrated the use of **JamSession** for the implementation of interaction protocols to coordinate two interacting work-

[5]Bonita is a comprehensive suite of open source software for workflow and business process management specification and execution. It can be found at http://www.bonitasoft.com/.

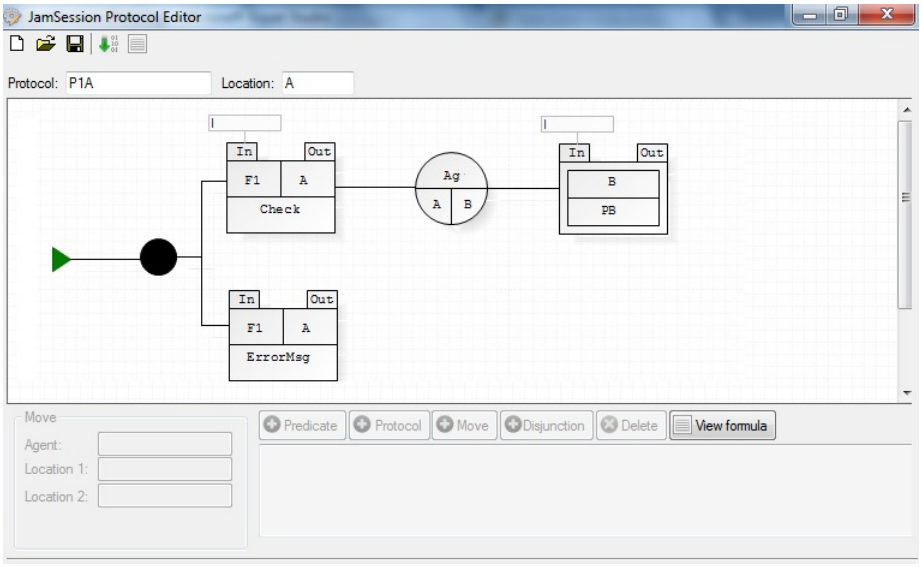

Figure 15: The kbip P_1^A in **JamSession** Protocol Editor.

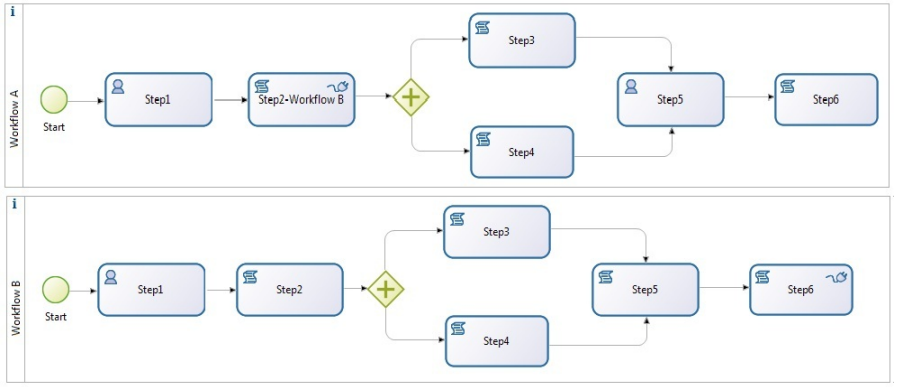

Figure 16: Example of Bonita workflow definitions with connectors.

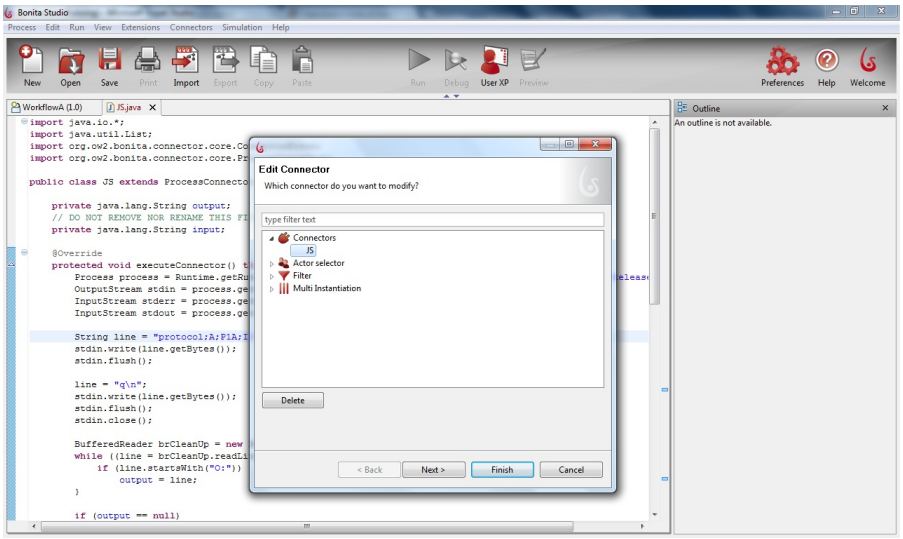

Figure 17: Bonita's connector for triggering the kbip P_1^A.

flows. Evidently, these ideas can be easily adapted to coordinate several workflows. For the general case, each organisation involved in a collaboration should have at least one special location in the graph, inhabited by a fixed special agent, who should be used to trigger general predicates such as *Check*, *ErrorMsg*, etc. The special locations of the participating organisations should be connected according to the flow of their messages, e.g. there should be an arc from location l_1 to l_2 if the organisation represented by l_1 sends at least one message to the organisation represented by l_2, and so forth for all participating organisations. In addition, several mobile agents should be used to represent the messages sent or received by the participating organisations, in such a way that moving such agents would indicate the triggering of a workflow instance (for the hierarchical pattern) or a message to be sent to an activity (for the peer-to-peer pattern). The number of agents would correspond to the maximal number of concurrent workflow instances that the organisation would coordinate. Hence, the agents should be returned to their original locations once the receivers completed the collaboration, i.e. after the sub-workflow or the activity had been successfully finished. Therefore, the locations should be connected by bidirectional arcs. Each organisation should use several auxiliary locations and agents for coordinating different workflows. In **JamSession** implementation, the agents can be distinguished by means of tags which can be thought intuitively as type identifiers.

The solution provided by **JamSession** to the problem of cross-organisational workflow collaboration is highly flexible and decoupled. Changes to the workflow structure do not affect the interactions as long as the connectors are preserved. On the other hand, only small changes are required to the connectors if the collaboration is updated. **JamSession** can coordinate workflows from different WfMS as long as the tools provide

means for the external management of workflow instances.

5. Verifying JamSession protocols for workflow collaboration

As shown in the previous section, **JamSession** allows decoupled interactions between workflows. The concept of locations inhabited by agents, and the movements of agents between locations are fundamental for this purpose. However, a careful management of agents is required since the orders to move agents may lead to errors which may be difficult to trace. Therefore, it is important to provide means to verify that the execution of the interactions meets the desired properties.

In this section we describe how **JamSession** protocols can be verified. As a tool to model the behaviour of a set of protocols and to investigate their dynamic properties, we use coloured Petri nets (Jensen (1992)). Petri nets can be effectively used to specify, simulate and formally analyse concurrent systems. Petri nets – and more specifically coloured Petri nets and their variations, such as hierarchical coloured Petri nets and timed coloured Petri nets – have also been used to specify, simulate and formally analyse specifically cross-organisational workflows (Jiang *et al.* (2010); Liu *et al.* (2011)). They have a compact graphical representation with several, well-defined primitives. As ordinary Petri nets (PNs), coloured Petri nets (CPNs) are directed bipartite graphs in which nodes are called places and transitions. As **JamSession** KBIPs, CPNs also have tokens which inhabit individual places and a set of net inscriptions (such as arc expressions and place initialisations). In addition, types (colour sets) are used in CPNs to describe the set of data values (colours) that a token can store. This way, each place has a type determining the kind of data (tokens) which the place may store. A place may contain several tokens with the same data value. The state of a CPN, i.e. the tokens which inhabit the places defined in the CPN, is called a marking. As for PNs, transitions represent actions or events. An incoming (respectively, outgoing) arc of a transition indicates that it may remove (respectively, add) tokens from the corresponding place. The exact number of tokens and their data values are determined by the arc expressions. The occurrence of a transition is conditioned to a binding of the variables in the expressions.

CPNs can be defined as hierarchical combinations of several non-hierarchical CPNs, using a hierarchy graph, which is defined as a multiset of CPNs, substitution transitions and fusion places. A substitution transition (also called supernode) allows to relate a transition and its edges to another CPN. Fusion places are sets of places which belong to different CPNs but are considered to be identical, i.e. places which are shared by two or more nets.

Each **JamSession** protocol can be modeled as a separate CPN. We use two colour sets $BOOL = \{\top, \bot\}$ and $LOC = \{L_1, L_2, \ldots, L_n\}$ (the set of locations shared by a collection of KBIPs). In addition, we use a function $s : LOC \times LOC \rightarrow BOOL$ to represent adjacency relations in the graph of locations. The hierarchical CPN has a single special place of type LOC to represent the state of the graph of locations, plus as many tokens as the number of agents which inhabit the graph of locations. The data value of each token coincides with the location of the corresponding agent in **JamSession**. We refer to this place as SGL.

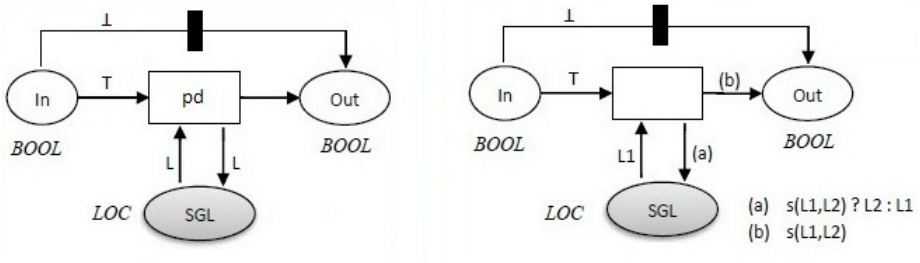

Figure 18: CPNs for $[a, L]Pred$ (left) and $move(a, L1, L2)$ (right).

The CPN associated to each **JamSession** entity has a unique input node (In) and a unique output node (Out) of type $BOOL$. Figure 18 shows the CPNs corresponding to a predicate call and to a move order. We employ the standard notation for CPNs, such that ellipses denote places and boxes denote transitions. In this paper, we use a black box to represent a transition whose outgoing arc expression is the same as the incoming arc. Note that during the execution of a workfow collaboration, all protocols share the same graph of locations. Therefore, in the CPNs of Figure 18, transitions associated to predicate calls and move orders may not be triggered as soon as a $\top$ token arrives at the input place. Due to the semantic of PNs, these transitions wait until the *SGL* place contains a token t with a particular *LOC* colour. Thus, they are not even guaranteed to occur. If they indeed do not occur, t is removed from the *SGL* and a token of *BOOL* colour is added to the output place. The value of the output token depends, respectively, on the predicate definition and the existence of the corresponding arc in the graph of locations. When a predicate call occurs, the content of the *SGL* place remains the same, and the removed token is returned to *SGL*. In the case of a move order, a new token of $L2$ colour is added. For brevity, we have used the ? and : operators for the arc expression instead of the more usual *if-then-else* construction.

This behaviour corresponds to the use of tokens as synchronisation elements. If the obtained semantics does not match with the intended behaviour of the KBIPs, then an alternative translation to CPNs is possible, in which the *SGL* place contains $SLOC = LOC \times int$ as colour set. It contains as many tokens as locations in the graph of locations, and the data value of each token characterises the location and ID of the corresponding agents in the KBIPs. The new CPNs are shown in Figure 19. However, when using this straightforward interpretation, the behaviour of a protocol collaboration may be unstable, since it may vary depending on the quantity and behaviour of users. Therefore, it is not appropriate for modeling the dynamic properties of concurrent protocols.

The remaining constructions allowed in **JamSession** can be modeled by means of substitution transitions. These transitions are used to relate a transition to more complex CPNs and are usually drawn using thick lines. The input/output place of a substitution transition must be related to the input/output place of the associated CPN. Figure 20 shows the CPNs corresponding to conjunctions and disjunctions. Substitution transitions A and B represent the corresponding CPNs in the expression. The

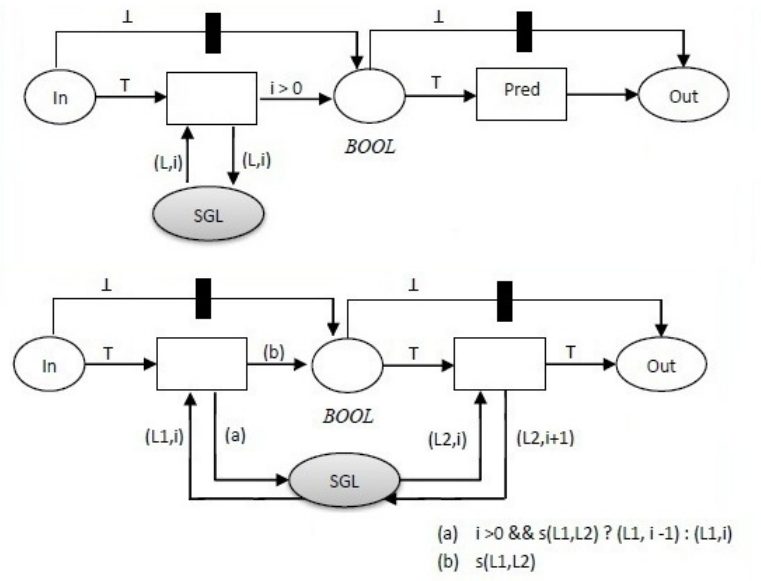

Figure 19: Alternative CPNs for predicate call (up) and $move(a, L1, L2)$ (down).

output token of the CPN for A enables an auxiliary transition which controls the activation of the CPN for B. The $empty$ constant indicates that the token should be removed, i.e. no token should be added to the inbound place of the arc.

The CPN associated to a protocol call have a substitution transition between the input and output places (see Figure 21 on the left). The supernode will represent the CPN corresponding to the protocol definition. The input/output place of the substitution transition belongs to the fusion set of the input/output place of the associated CPN. Therefore, when a token is added/removed at the input/output place of the transition, it is also added/removed to/from the input/output place of the CPN. Figure 21 (right) shows the use of a fusion set in a compact CPN for a conjunction.

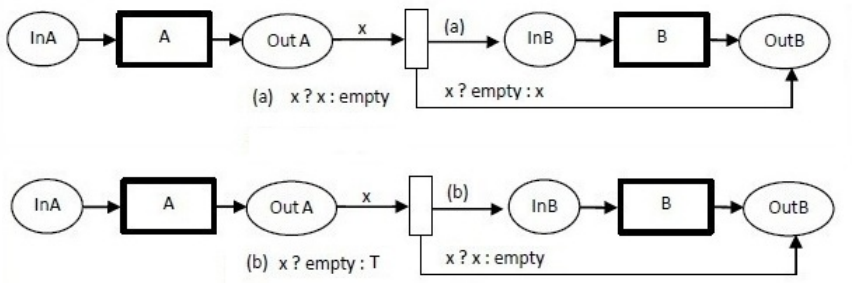

Figure 20: CPNs for a $A \wedge B$ (up) and $A \vee B$ (down) formulas.

Figure 21: CPN for a protocol call (left) and alternative CPN for $A \wedge B$ (right).

When a set of **JamSession** protocols defines a hierarchy (as in workflow collaboration), a non-hierarchical CPN can be obtained by replacing each supernode by an instance of the corresponding CPN. The construction can be done top-down, bottom-up or mixing both strategies. Figure 22 depicts the hierarchical CPN associated to the set of protocols defined in Section 4 for the *hierarchical pattern* (Figures 7, 8 and 9). The places with the same label belong to the same fusion set. Without loss of generality we have assumed that fixed agents belong to special locations. The interaction resulting from the *peer-to-peer pattern* can be modeled by a linear sequence of asynchronous messages between the participants. For the sets of protocols presented in Section 4, this corresponds to the formula $[A]QA1 \wedge [A]QB2$. Figure 23 depicts the hierarchical CPN associated to $[A]QA1$. The hierarchical CPN associated to $[A]QB2$ is analogous. The output place of an instance of former will be fused with the input place of an instance of the latter to obtain the hierarchical CPN corresponding to $[A]QA1 \wedge [A]QB2$.

It is not difficult to see that the behaviour of a **JamSession** formula corresponds to the behaviour of the related CPN for an initial marking having a token with $\top$ colour at the source place[6] of the net and the SGL place with the initial state of the graph of locations. This way, all analysis tools for CPNs can be used for hierarchical protocols (even without building the translation to a non-hierarchical CPN). Properties like reachability and boundedness can be used to analyse the behaviour of KBIPs. For instance, the verification at the end of Section 4.1 can easily be obtained using a PN tool. This is done by computing the reachability tree which is finite and checking that all leaves have a single $\top$ token at the sink place of the net. The rest of the places in the net should be empty, except for the SGL which has the final configuration of the graph of locations. This condition guarantees the termination of the interaction without conflicts or deadlocks. In addition, if the final configuration of the graph coincides with the initial one or meets some specific requirements of the workflow collaboration, then the behaviour of the interaction is correct. By analysing the reachability tree we can also compute the best possible upper and lower bounds for the places in the net, and hence the minimum and the maximum number of agents required for the interaction.

6. Discussion and Future Work

In the present article we have discussed the issue of designing and executing cross-organisational workflow integration, and we have focused primarily on the execution of workflow integration. To this end, we have introduced the **JamSession** coordina-

[6] A source node has no ingoing arc while a sink node has no outgoing arc. In Figure 22 the source place is In while the sink place is Out.

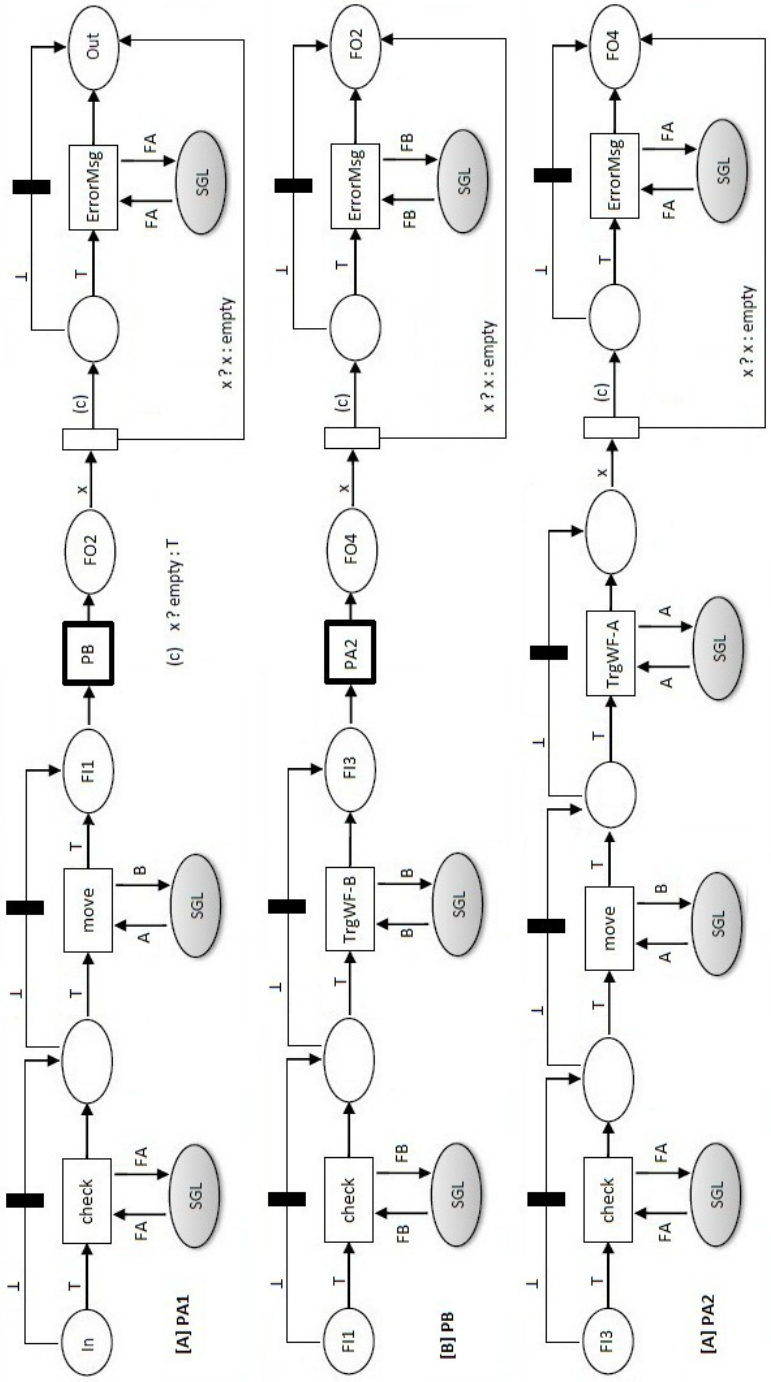

Figure 22: Hierarchical CPN for a hierarchical workflow pattern.

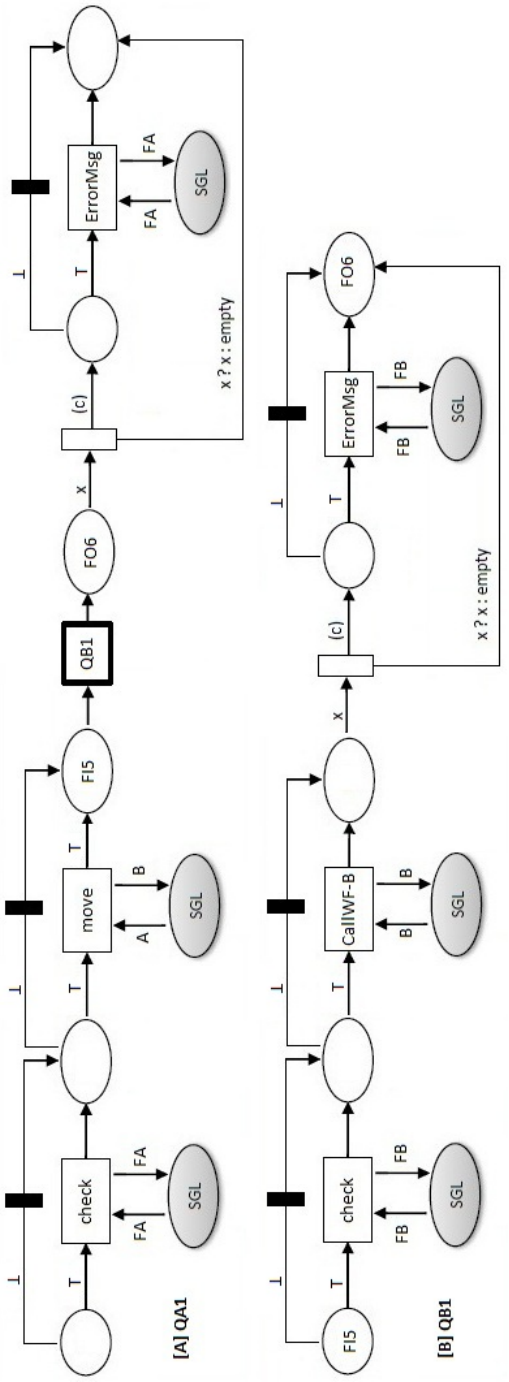

Figure 23: Hierarchical CPN for a peer-to-peer workflow pattern.

tion platform, and illustrated how it can be employed effectively to implement cross-organisational workflow integration.

JamSession is a very lightweight and user-friendly platform which provides a suitable framework for specifying and executing cross-organisational workflow interactions. Using this platform, workflows and activities are modeled by means of knowledge-based interaction protocols and predicates, in such a way that these definitions are local to each workflow management system and just the interaction protocols are made public.

The internal processes of a workflow remain hidden for the partners involved in a workflow collaboration. Thus, each partner may change its private workflow process as long as the protocol is not affected. The concept of locations, agents and movements between locations are fundamental for synchronizing groups of protocols. The coordination is performed in a simple, flexible and decoupled way. Hence, **JamSession** is an effective platform to implement interactions between existing workflows. As **JamSession** can interact with different workflow specification and execution engines, it can be a powerful and flexible tool for cross-organisational workflow implementation.

The declarative semantics of KBIPs in **JamSession** provides the means for formal analysis and verification of protocols. In particular, the translation of protocols to coloured Petri nets can be used to simulate the behaviour of an interaction and to analyse the possible conflicts, faults and exception states. Based on these analyses, optimised KBIPs can be built, in which agents can be treated more effectively and the final outcomes correspond to the intended behaviour of the collaboration. In general the Petri nets approach seems a promising line for studying the dynamic properties of sets of **JamSession** protocols. This is the subject of ongoing research and shall be described in future publications.

References

L. Biegus and C. Branki. India: a framework for workflow interoperability support by means of multi-agent systems. *Engineering Applications of Artificial Intelligence*, 17(7), 2004.

A. Bogdanovich, M. Esteva, S. Simoff, C. Sierra, and H. Berger. A methodology for 3d electronic institutions. In *6th International Joint Conference on Autonomous Agents and Multiagent Systems*, USA, 2007.

A. Byde, G. Piccinelli, and W. Lamersdorf. Automating negotiation over b2b processes. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, 2002.

I. Chebbi, S. Dustdar, and S. Tata. The view-based approach to dynamic interorganizational workflow cooperation. *Data and Knowledge Engineering*, 56(2), 2006.

Xi Chen and P. W. H. Chung. Cross-organisational workflow enactment via progressive linking by run-time agents. In *Advances in Applied Artificial Intelligence*, Germany, 2006.

Xi Chen and P. W. H. Chung. A framework for cross-organizational workflow collaboration. In *Proceedings of 13th Cross-strait Academic Conference on Information Management Development and Relevant Strategy*, 2007.

Q. Chen and M. Hsu. Inter-enterprise collaborative business process management. In *Proceedings of 17th International Conference on Data Engineering (ICDE-01)*, Germany, 2001.

Xi Chen and L. Yang. Applying ai planning to semantic web services for workflow generation. In *Proceedings of the First International Conference on Semantics, Knowledge and Grid*, USA, 2005. IEEE Computer Society.

F. Chen, C. Ren, J. Dong, Q. Wang, J. Li, and B. Shao. Modeling cross-organizational services composition with pi-calculus. In *Proceedings of the IEEE International Conference on Service Operations, Logistics, and Informatics (SOLI)*, 2011.

Xi Chen. *IT Supported Business Process Negotiation, Reconciliation and Execution of Cross-organisational e-Business Collaboration*. PhD thesis – Loughborough University, UK, 2008.

F. S. Correa da Silva and W. Vasconcelos. Rule schemata for game artificial intelligence. In *Proceedings of the Joint Ibero-American / Brazilian Symposium of Artificial Intelligence (IBERAMIA/SBIA)*, Brazil, 2006. Springer LNAI.

F. S. Correa da Silva. Knowledge-based interaction protocols for intelligent interactive environments. Knowledge and Information Systems – accepted, 2011.

R. Eshuis and P. Grefen. Constructing customized process views. *Data and Knowledge Engineering*, 64(2), 2008.

M. Esteva, J. A. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. L. Arcos. On the formal specification of electronic institutions. In *Agent Mediated Electronic Commerce – The European AgentLink Perspective*, UK, 2001. Springer LNCS.

P. Grefen, N. Mehandjiev, G. Kouvas, G. Weichhart, and R. Eshuis. Dynamic business network process management in instant virtual enterprises. *Computers in Industry*, 60(2), 2009.

N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4), 2001.

K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, Berlin, 1992.

P. Jiang, X. Shao, H. Qiu, and P. Li. Interoperability of cross-organizational workflows based on process-view for collaborative product development. *Concurrent Engineering: Research and Application*, 16(1), 2008.

P. Jiang, L. Gao, P. Li, and H. Qiu. Collaborative execution mechanisms for the tcpn enhanced process-view approach based inter-enterprises workflow. In *Proceedings of the 13th International Conference on CSCW in Design (CSCWD 2009)*, 2009.

P. Jiang, X. Shao, L. Gao, H. Qiu, and P. Li. A process-view approach for cross-organizational workflows management. *Advanced Engineering Informatics*, 24(2), 2010.

D. Krafzig, K. Banke, and D. Salma. *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice Hall PTR, USA, 2004.

D. Krukkert. Matchmaking of ebxml business processes. In *Technical Report IST-28584-OX-D2.3-v.2.0 – openXchange Project*, 2003.

Y. Liu, Y. Shen, and T. Hao. Research on reliability modeling of cross-organizational workflows based on hierarchical colored petri nets. *Advanced Materials Research*, 186, 2011.

V. R. Pratt. Modelling concurrency with geometry. In *Proceedings of the 18th Annual ACM Symposium on Principles of Programming Languages*, 1991.

D. Robertson. A lightweight coordination calculus for agent systems. In *Proceedings of Declarative Agent Languages and Technologies*, USA, 2004. Springer LNCS.

M. Saleem, P. W. H. Chung, S. Fatima, and W. Dai. Intelligent business transaction agents for cross-organizational workflow definition and execution. In *Proceedings of Intelligent Information Processing V (IFIP Advances in Information and Communication Technology*, UK, 2010.

M. Saleem, P. W. H. Chung, S. Fatima, and W. Dai. Cross organisational compatible plans generation framework. In *Proceedings of AI-2011 Thirty-first SGAI International Conference on Artificial Intelligence*, 2011.

M. Saleem. *Cross Organisational Compatible Workflows Generation and Execution*. PhD thesis – Loughborough University, UK, 2012.

K. A. Schulz and M. E. Orlowska. Facilitating cross-organizational workflows with a workflow view approach. *Data and Knowledge Engineering*, 51(1), 2004.

E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of web services using semantic descriptions. In *Proceedings of Web Services: Modelling, Architecture and Infrastructure Workshop (in conjunction with ICEIS)*, 2003.

E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. Htn planning for web service composition using shop2. *Journal of Web Semantics*, 1(4), 2004.

W. M. P. van der Aalst. Interorganizational workflows: an approach based on message sequence charts and petri nets. *Systems Analysis – Modelling – Simulation*, 34(3), 1999.

W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman. Normative conflict-resolution in multi-agent systems. *Autonomous Agents and Multiagent Systems*, 19(2), 2009.

G. Vizzari. *Dynamic interaction spaces and situated multi-agent systems: from a multi-layered model to a distributed architecture*. PhD thesis – University of Milano-Bicocca, Italy, 2004.

S. Y. Wang, W. M. Shen, and Q. Hao. An agent-based web service workflow model for inter-enterprise collaboration. *Expert Systems With Applications*, 2006.

WfMC. *The Workflow Reference Model – Technical Report WFMC-TC-1003*. Workflow Management Coalition, 1995.

J. Yang and M. Papazoglou. Interoperation support for electronic business. *Communications of the ACM*, 43(6), 2000.