Norta *et al. Journal of Internet Services and Applications* (2015) 6:8
DOI 10.1186/s13174-015-0023-7

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

eContractual choreography-language properties towards cross-organizational business collaboration

Alex Norta[1*], Lixin Ma[1,3], Yucong Duan[2], Addi Rull[1], Merit Kõlvart[1] and Kuldar Taveter[1]

Abstract

Meaningfully automating sociotechnical business collaboration promises efficiency-, effectiveness-, and quality increases for realizing next-generation decentralized autonomous organizations. For automating business-process aware cross-organizational operations, the development of existing choreography languages is technology driven and focuses less on sociotechnical suitability and expressiveness concepts and properties that recognize the interaction between people in organizations and technology in workplaces. This gap our suitability- and expressiveness exploration fills by means of a cross-organizational collaboration ontology that we map as a proof-of-concept evaluation to the eSourcing Markup Language (eSML). The latter we test in a feasibility case study to meaningfully support the automation of business collaboration. The developed eSourcing ontology and eSML is replicable for exploring strengths and weaknesses of other choreography languages.

Keywords: Smart contracting; Choreography; eSourcing; Suitability; Expressiveness; Cross-organizational; B2B; Business process; Sociotechnical; Decentralized autonomous organizations

1 Introduction

With the emergence of new automation paradigms such as service-oriented computing (SOC) and cloud computing (CC), the way companies collaborate with each other experiences significant changes. SOC [1] comprises the creation of automation logic in the form of web services. In CC [2], access to web-based applications, web services, and IT infrastructure as a service happens through the Internet. Web services [3] are an important vehicle for enabling organizations to cooperate with each other by cross-organizationally linking business processes [4-7] with choreography languages for the purpose of electronic outsourcing. More recently, a trend-reinforcement occurs with so-called decentralized autonomous organizations and -corporations that are powered by smart contracts [8,9] to form agreements with people via the block chain [10]. The ontological concepts and properties for the design of smart-contracting systems [11] we derive from legal principles, economic theory, and theories of reliable and secure protocols. The smart contract itself is a computerized transaction protocol [12] that executes the terms of a contract. The blockchain is a distributed database for independently verifying the chain of ownership of artefacts in hash values that result from cryptographic digests [13].

With respect to existing choreography languages, the most notable are versions of the Business Process Execution Language such as AbstractBPEL [14] and BPEL4Chor [15], Web Services Choreography Description Language (WS-CDL) [16], Business Process Modeling Notation (BPMN) [17,18] Let's Dance [19], ebXML BPSS [20] and more recently, the Business Choreography Language (BCL) [21]. However, not only existing choreography languages but also other XML-based languages for SOC lack adoption by industry. A reason is the approach for language development that does not take into account sociotechnical suitability and expressiveness deficiencies that recognizes the interaction between people in organizations and technology in workplaces. Sociotechnical systems comprise theory about the social aspects of people and society and technical aspects of organizational

*Correspondence: alex.norta.phd@ieee.org
[1]Tallinn University of Technology, Akadeemia Tee 15A, 12618 Tallinn, Estonia
Full list of author information is available at the end of the article

structure and processes. Suitability means that choreography languages comprise concepts and properties to allow the formulation of real-world business-collaboration scenarios in many perspectives. Expressiveness means the constructs of a choreography language have semantic clarity for ensuring uniform enactment behaviour by different business process engines. Additionally, contractual agreements are the foundation of business collaboration. This paper fills the gap by answering the research question how to systematically develop a language for cross-organizational and contract-based collaboration specifications. From there we deduce several sub-questions. What is the collaboration context and model the specification language must cater for? What are the main suitability- and expressiveness concepts and -properties? The means to answer the research questions are first, a so-called eSourcing ontology [22] that comprises the concepts and properties we generate from the suitability and expressiveness study. The eSourcing ontology development with the tool Protégé [23] allows for an application of the HermiT OWL reasoner [24] to check for the ontology consistency, identify subsumption relationships between classes, and so on. Secondly, as a means of feasibility evaluation, the eSourcing ontology we translate into the eSourcing Markup Language eSML for which we give the schema definition [25] and additional documentation online. The core difference in the approach to developing eSML is the existence of process views [4] for respective collaborating organizations for establishing a contractual consensus during the collaboration-setup phase. These process views are subsets of larger in-house processes of which extensions remain opaque to the counterparty to protect privacy, business secrets, and so on.

The objective is to enable contractual flexibility as significant changes in integral business processes must be enabled by a high degree of automation. Traditionally, if something will happen in the future, then a contract must include rules that regulate these particular instances. Usually, most of regulation is unnecessary since presupposed events never occur. Therefore, automation requires taking a deductive approach towards contracting by exploring relevant concepts with a focus on the basic contractual elements only. It is sufficient to determine only the concepts and properties without which a transaction cannot be executed [26]. In general, basic contractual elements comprise of parties, offer, acceptance, rights and obligations. Furthermore, the reduction of contractual elements helps to overcome difficult legal issues such as what law to apply, the legal context, how to determine rights and obligations. The common source of reference to the basic contractual elements is not any national law, but lex mercatoria [27] and supranational model rules, e.g., PECL[a], PICC[b], DCFR[c], CISG[d], Incoterms[e].

The structure of the paper follows the design-science method [28,29] for the development of eSML and is as follows. Section 2 presents a business-collaboration model that evolves from case studies in the research project called CrossWork [30,31], namely eSourcing [4,7,32,33]. In Section 3, we further explore the collaboration model in a pattern-based way [32] with the objective of generating the essential concepts for eSML to gain business-collaboration suitability. Next, assuming the control-flow perspective is dominant for enacting business collaborations, we present in Section 4 the expressiveness-assurance in eSML. Section 5 presents in a feasibility evaluation the resulting structure of eSML and shows examples, followed by discussing a "proof-of-construction" application system. Section 6 gives related work and finally, Section 7 concludes this paper and discusses future work.

2 Business collaboration model

In the EU research project CrossWork [30,31], observing business collaborations of industry partners reveals characteristic features. An original equipment manufacturer (OEM) develops value chains in an in-house business process according to different perspectives, e.g., control flow of tasks, information flow, personnel management, allocation of production resources, and so on. The CrossWork case studies [31] reveal that the basis for business collaboration between organizations are contracts. The basis has implications for the suitability exploration in the sequel. Next, Section 2.1 explains the orthogonal collaboration dimensions of client/server versus peer-to-peer (P2P). Section 2.2 discusses the structural properties of the client/server collaboration model. Finally, Section 2.3 shows how the collaboration model with the same structural properties also enables P2P-collaboration when the roles of the collaboration-elements change.

2.1 Collaboration dimensions

Figure 1 conceptually depicts conceptually a complex collaboration scenario of an OEM with suppliers. The reasons for acquiring services externally are manifold, e.g., the OEM can not produce with the same quality, or an equally low price per piece, the production capacity is not available, required special know-how is lacking, and so on.

The horizontal ellipses in Figure 1 denote the client/server-integration of outsourced in-house process parts to lower-level clients who provide services to the vertically adjacent higher tier of a supply chain [32]. The outsourced business processes are refined with additional process steps by the respective suppliers. The refinements remain opaque to the service consumer and the supplier only has awareness of the OEM's outsourced respective process but the remaining in-house process remains

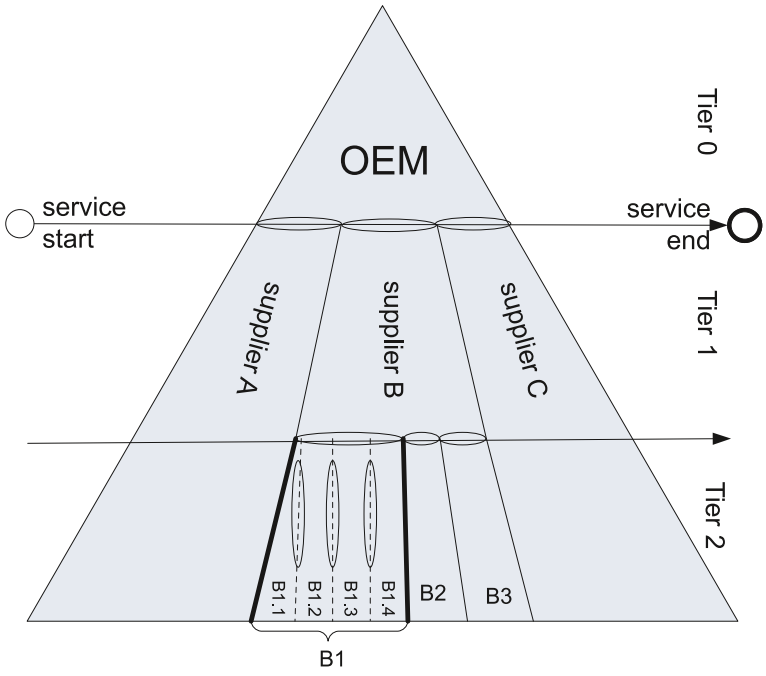

Figure 1 B2B-collaboration dimensions.

opaque. For client/server-integration, several projects investigate enterprise interoperability [34-36].

Vertical ellipses in Figure 1, depict a P2P-collaboration within a cluster of small and medium sized enterprises (SME). If several SMEs form a composed service in a P2P way [37], they together turn into a supplier for a higher-level service consumer. In this paper, we consider the vertical collaboration between organizations by the means of service choreography for the subsequent suitability and expressiveness exploration.

2.2 Client/server-collaboration model

As an explanation of vertical business collaboration, Figure 2 depicts a three-level model as part of an eSourcing example [4,7,32]. The three-level model is instrumental for not forcing collaborating parties into connecting their information infrastructures directly. The processes in Figure 2 depict the control-flow perspective of the eSourcing concept that focuses on structurally harmonizing on an external level the intra-organizational business processes of a service consuming and one or many service providing organizations into a business collaboration. Important elements of eSourcing are the support of different visibility levels of corporate process details for the collaborating counterparts and flexible mechanisms for service monitoring and information exchange. Recently, leading IT-enterprises launched eSourcing [38] application systems [39,40] to enable business collaboration and in [33] we evaluate these systems against the eSourcing Reference Architecture eSRA.

The very top and bottom of Figure 2, show the internal levels of the service consumer and -provider respectively where processes are directly enactable by legacy systems, which caters towards a heterogeneous system environment, e.g., by workflow management systems. Furthermore, processes of the OEM and service providers on a conceptual level are independent from infrastructure and collaboration specifics. In the center of Figure 2, the external level stretches across the respective domains of eSourcing parties where structural process matching takes place and for which eSML is applicable. Either collaborating counterparties project only interfaces, or parts, or all of the respective conceptual-level processes to the external level for performing business-process matching [4,7]. A contractual consensus between collaborating parties comes into existence when the projected processes are matched externally, i.e., when they are equal. Not projected process parts remain opaque to the collaborating counterparts.

More recently, research in [4] demonstrates with BPMN and BPEL the feasibility of this approach with industry standards. The eSourcing model in Figure 2 shows we use Petri-net formalism for exploring structural properties [7]. The dashed monitoring arcs [32] in Figure 2 connect the conceptual business processes via the external level into a configuration. In Section 4, we expand on the structural properties of eSourcing configurations.

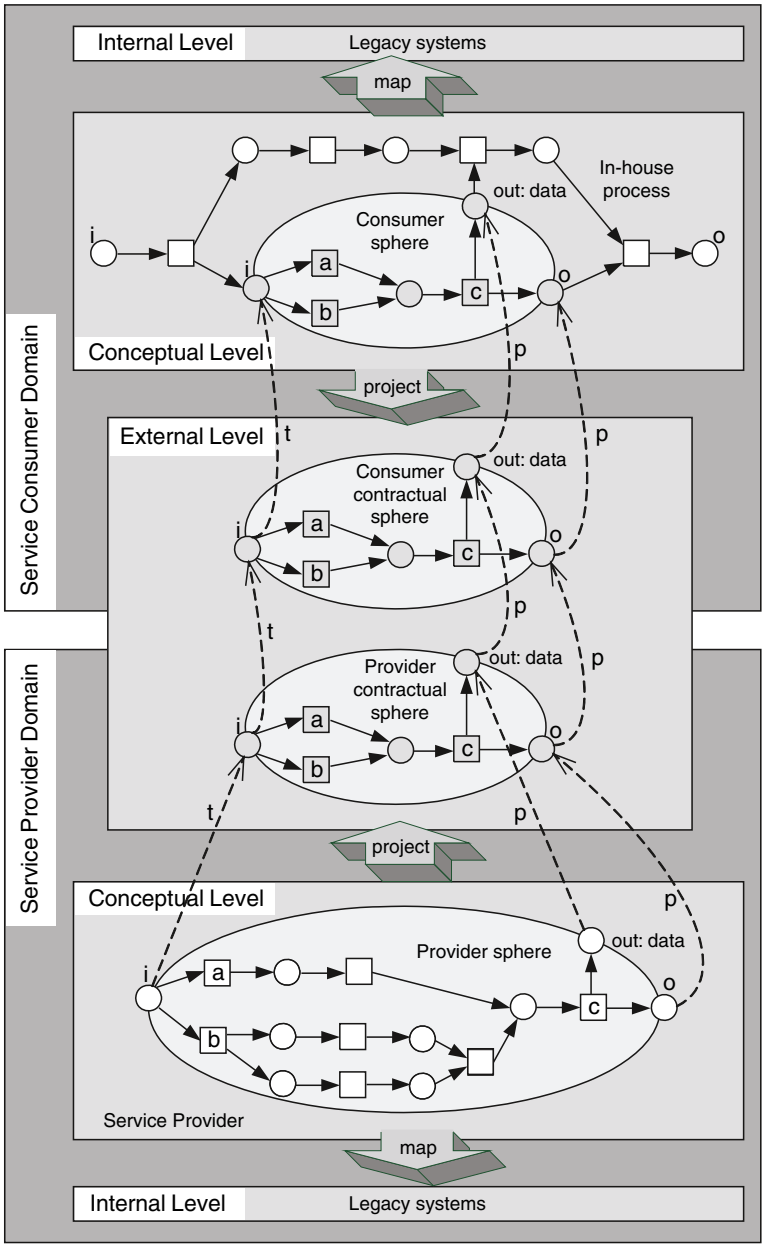

Figure 2 eSourcing in three-level business-process as master/client-collaboration.

2.3 P2P-collaboration model

With the same structural properties as explained above related to Figure 2, also P2P-collaboration are possible when the roles of the collaboration-model elements change. Figure 3 depicts these changed roles conceptually.

In comparison to the master/client-collaboration of Figure 2, the in-house process of a service consumer is a so-called business-network model (BNM) [41] in the P2P-case. A BNM captures choreographies that are relevant for a business scenario. A BNM contains legally valid template contracts that are service types with assigned roles. Together with the BNM, the service types with their roles are available in a collaboration hub that houses business processes as a service (BPaaS-HUB) [42] in the form of subset process views [4]. The latter addresses the need to semi-automatically find collaboration parties and learn about their identity, services, and reputation. A BPaaS-HUB enables speedy business-partner discovery and support for on-the-fly background checking with a matching of services.

On the external layer of Figure 2, now service offers match with service types from the BNM identically to

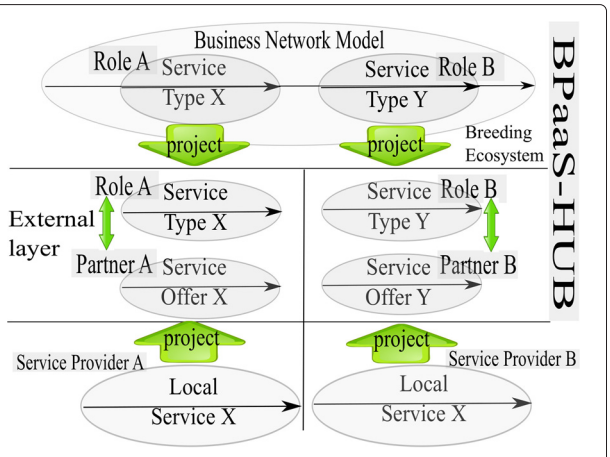

Figure 3 P2P-collaboration with eSourcing.

the contractual sphere of collaborating parties in Figure 2. Additionally, a collaborating partner must match into the role specifics associated with a respective service type. The matching must follow the same structural rules as expressed in Section 2. Likewise, the local services of the service providers in Figure 2 are behavioural sub-classes to their corresponding service offers the same way as the provider's contractual sphere relates to the provider sphere. In the sequel, we discuss the structural properties such collaboration configurations must adhere to.

Next, we explore the required set of concepts and properties for specifying a contractual electronic choreography-specification language.

3 Ontological suitability exploration

There are two angles for approaching the suitability exploration. First, in Section 3.1 we explore cross-organizational collaboration from the paradigm that contracts are the foundation. This is the differentiating approach to choreography development that we ontologically explore by generating the HermiT-OWL reasoner [43] checked eSourcing ontology first that we translate as a means of feasibility evaluation into eSML secondarily. For downloading, we provide links in Section 1. In Section 3.2, a pattern-based exploration further details the contractual collaboration paradigm that are again input for the eSourcing ontology- and eSML development.

3.1 eContract-based exploration

For ensuring that the eSourcing ontology and the subsequently deduced eSML comprises sociotechnial concepts to allow the formulation of real-world business-collaboration in relevant perspectives like control-flow, data-flow, resources and so on, the case-study findings culminating in the eSourcing model of Figure 2, require more exploration. Taking pre-existing work about contract automation [44] into account, we extend the set of concepts and properties for the eSourcing ontology and eSML to achieve suitability in accordance with Section 2 where we deduce features from the business collaboration model.

The eSourcing ontology we base on a smart contracting foundation [9] namely the XML-based language ECML (*E*lectronic *C*ontracting *M*arkup *L*anguage) [44]. Thereby, the latter is also incorporated into eSML that we use for the feasibility evaluation in the sequel. A smart contract is a legally enforceable agreement in which two or more parties commit to certain obligations in return for certain rights [45,46]. Contracts are instruments for organizing business collaborations. Smart contracting aims at using information technologies to significantly improve the efficiency and effectiveness of paper contracting, allowing companies to support newly emerging business paradigms, while still being legally protected.

Although ECML permits business-process definitions, it lacks a clear collaboration-model support as proposed by eSourcing. Inheriting concepts from ECML, at the highest abstraction level, a contract in the eSourcing ontology and eSML answers three conceptual questions i.e., the Who, Where, and What question for which we refer the reader to [44] for further details.

3.1.1 *The Who concept*

This concept that we depict in Figure 4, legally clearly identifies the contracting parties by including the class `party`. Parties are actors that have rights and obligations that are listed in the eSourcing configuration. Concerning the relationship cardinalities, it is defined that at least two party specifications must be part of a contract. It is also possible to have more than two parties defined. For example, an original manufacturer can agree with several suppliers to be part of one contract.

In a contract, several third parties termed `mediators` may optionally be part of an electronic contract. Mediators represented by class `mediator` participate in the enactment of an eSourcing configuration without statements about rights/obligations. Consequently, mediators do not have to sign the eContract. If their relations definition with the parties is legally binding, the mediators become a party in the same, or in a separate contract that states their rights and obligations. For example, a mediator verifies whether an eSourcing configuration that is part of a contract terminates successfully from a control-flow point of view. In order to safeguard business details from each other, the contracting parties are not allowed to check such correct termination themselves without disclosing their business secrets to each other.

Contracting parties and optional numbers of mediators are in a relationship with several other classes.

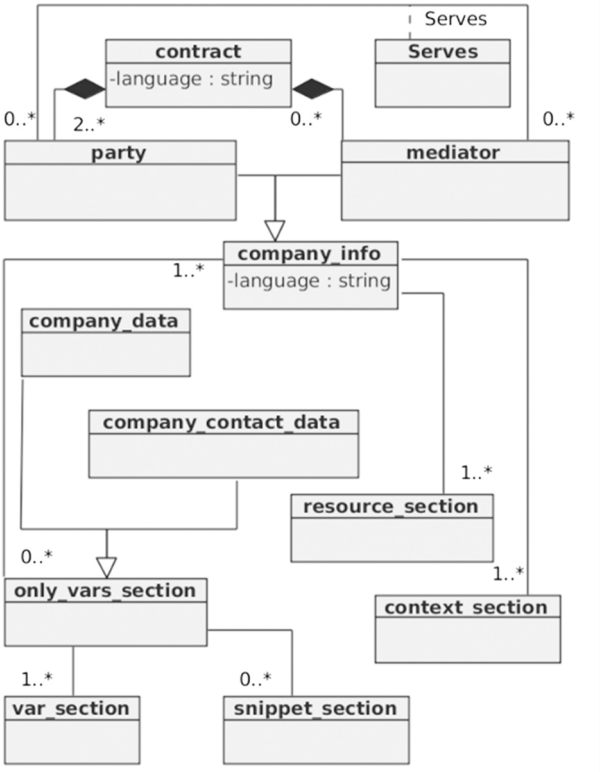

Figure 4 The `Who` model.

The `company_data` comprises, e.g., the name of a contracting party or mediator, the type of legal organization, and so on. The `company_contact_data` refers to the geographic location of an eSourcing party to uniquely identify according to legal requirements a contracting party, or a mediator.

The class `resource_section` is the root of the resource perspective comprising a contracting `party`, or a `mediator` with optionally attached resource definitions. However, the latter is superfluous unless a mediator is part of commercial exchanges. Resources are actors and non-actors of which the latter is either consumable, or non-consumable.

The classes `company_data` and `company_contact_data` are subclasses of class `only_vars_section` that contains variables and so-called process snippets. The class `var_section` is a connection to the data-flow perspective that includes company description, trade registration number, VAT registration number, address of registration, etc. The class `snippet_section` references so-called contract snippets that are attachable to particular contract definitions, e.g., to attach general terms and conditions. We refer to [25] for details about the resource perspective that is part of the *Who*-concept.

3.1.2 The Where concept

In Figure 5, we distinguish two basic aspects of the electronic contracting context, i.e., the business context and the legal context. Thus, the Where-section comprises two separate parts. In addition, a third subsection optionally references other electronic contracting provisions that are not part of the core legal and business context.

All three classes named `business_context_provisions`, `legal_context_provisions`, and `other_context_provisions` are subclasses of the grouping class named `all_section`. It references the classes `process_section`, `var_section`, `rule_section`, and `snippet_section`.

3.1.3 The What concept

As depicted in Figure 6, the What-model contains concepts related to the exchanged values and their related conditions. Two main subsections of the What-concept are the `exchanged_value` and the corresponding `exchange_provisions` for the value exchange. These classes are defined separately for every respective contractual party involved in contracting.

In a case of product exchange, the product description employs data constructs. In a case of service exchange, the service description combines data-flows and process constructs. The corresponding financial reward for the received value (in non-barter exchanges) uses the same constructs as a service description subsection. The value-exchange provisions subsection requires the use of rule and process-specification constructs. Examples for exchange provisions are rules for determining how late payment needs to be handled, how cancellations are dealt with, or definitions for calculating interest adjustments in payments.

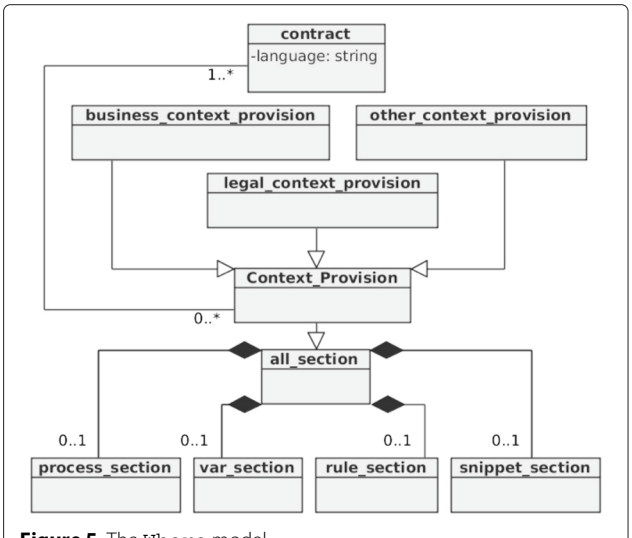

Figure 5 The `Where` model.

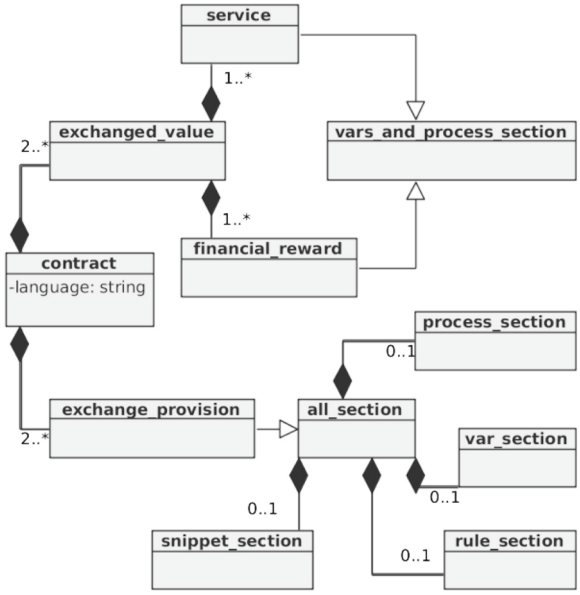

Figure 6 The `What` model.

The Who-, What-, and Where concepts respectively correspond to the concepts of agents, the knowledge by agents, and the agents' environment respectively. We understand agents as autonomous entities situated in an environment that perceive events occurring in the environment, reason based on knowledge, and act on the basis of perceived events. The concept of agents and related concepts are crucial in negotiating and achieving contracts within sociotechnical systems as contracts can only occur between autonomous entities that should be conceptually understood as agents. We refer to [47] for a detailed discussion.

The concept of agents and related concepts are particularly relevant for the P2P-collaboration model within a cluster of SMEs overviewed in Section 2.3 as SMEs are autonomous entities. According to the P2P-collaboration model, several SMEs form a composed service in a P2P way and this way jointly become a supplier for a higher-level service consumer. The P2P-collaboration model between autonomous SMEs is partially, or fully automated [47] by software agents that negotiate and conclude contracts on behalf of the enterprises. The software agents need the eSML for representing the contracts.

Next, we further explore the suitability features of electronic contracting in using patterns that are conceptual and on the outset technology-agnostic.

3.2 Pattern-based exploration

The chosen method for continued suitability exploration of additional business-collaboration concepts is as follows. To translate the eSourcing model of Figure 2 into a suitable ontology and subsequently, a choreography language, we deduce several feature dimensions in the form of axes that create a multi-dimensional, logical space. On every axis, dimension values detail the eSourcing feature an axis represents. By taking a subset of axes, we create a logical space that represents a particular eSourcing perspective. Consequently, the axes and their contained values serve as a taxonomy for ordering and relating to each other a set of perspective-relevant patterns. Note, we present a high-level overview of the pattern space and refer to [32] for the actual pattern specifications.

The three axes in Figure 7 represent different eSourcing dimensions with values. The created multi-dimensional space is instrumental for deducing eSourcing-construction elements for protecting internal business details, ensuring data exchange that adheres to correct control-flow, and for permitting the service consumer a controlled observation of the service provider's enactment progress. Correspondingly, the axes of the multi-dimensional space of Figure 7, represent the conceptual dimensions called contractual visibility, conjoinment, and monitorability [32]. The first conceptual dimension permits deducing interaction patterns [25] that occur during the setup phase of an electronic business collaboration. The interaction patterns are input for the proof-of-construction prototype we give in Section 5.4. The latter two conceptual dimensions of Figure 7 turn into eSML language constructs in the sequel.

The cube dimensions and values of Figure 7, are as follows. Contractual visibility focuses on the amount of business-process nodes a collaborating party projects to an external level to be visible for the counterparty. First, a *white-box* value means all nodes of a process part to be sourced are externalized. In case of a *black-box* value, only the interfaces of that process part are projected. Finally, the *gray-box* value means, the interfaces and a subset of the nodes and arcs of the externally sourced process part are projected.

Conjoinment focuses on the exchange of business information between the domains of the collaborating parties. Consequently, the business processes within the domains contain equal conjoinment constructs. *One-directional* conjoining implies that there is one *out*-, or *in*-directed information exchange between the domains of a service consumer and provider. *Bi-directional* conjoining is initiated by an *out*-directed information exchange to the domain of the collaborating counterpart who returns the communication exchange immediately to the initiating party.

Monitorability covers the way how nodes in the consumer's and provider's conceptual-level business processes link to each other via constructs with the properties termed *messaging* and *polling*. The nodes of the externalized process part connect to nodes in the corresponding

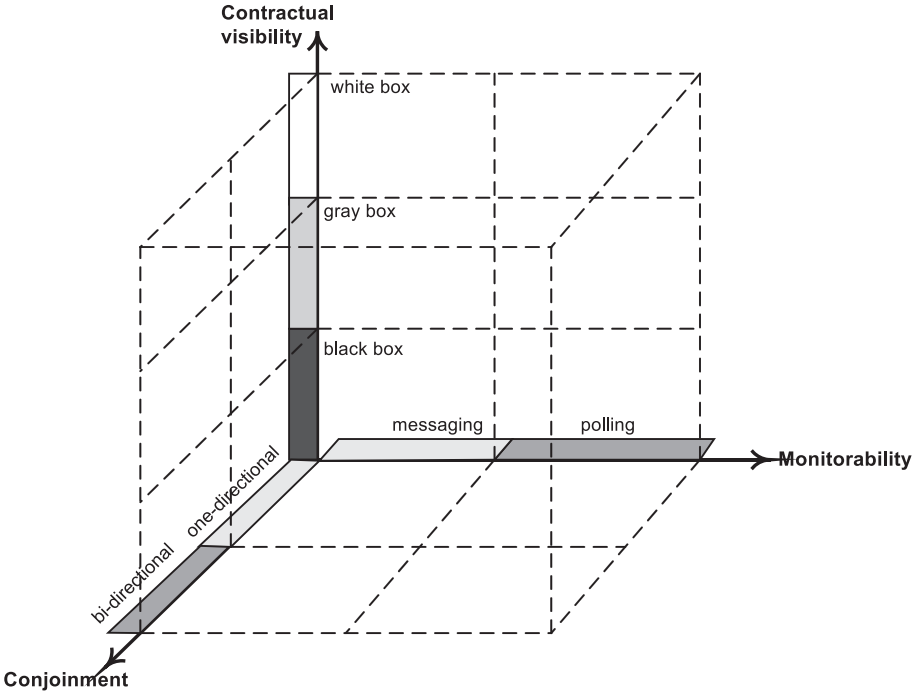

Figure 7 Dimensions and values of the eSourcing perspective.

service-provider process. The degree of monitorability of service provisioning for a service consumer increases by the amount of node linkages. At a minimum, all interface nodes of both domain processes need to be linked with each other. Additional nodes may be linked that belong to the respective business processes of service consumers and service providers. We refer to [32] for detailed pattern specifications and to [25] for collaborating counterparty-interaction patterns during setup.

The remainder of this section shows concepts in models that are instrumental for supporting the logical pattern space.

3.2.1 Data-package model

In an eSourcing configuration, a contract defines variables and documents that are relevant for enactment. Figure 8 shows entities for integrating such data into an eSourcing configuration. A `contract` references a `data_definition_section` that in return references one or several `data_package` instances. These data packages optionally contain a set of variables and documents.

It must be possible to reference data by other electronic contracting elements. It is essential for safeguarding that data have specific types to allow for assured contract processing. Data types set basic constraints on the allowed values for a data element. Two classes of required data types we identify, namely standard- and special data types. The identified data types we adopt from ECML and in [44] further details and examples can be found.

3.2.2 Process model

In Figure 9, the process model contains classes that belong to the control-flow perspective. A `process_section` may contain no or multiple process definitions. A process is a type of route, which is the root class for a process definition. All remaining classes of the `process_section` in [25] are part of the eSourcing perspective. The majority of those classes are for defining and mapping the life-cycles of service consumer- and provider processes that are involved in an eSourcing configuration.

The class `life_cycle_definition` is optionally multiple times part of `process_section` and

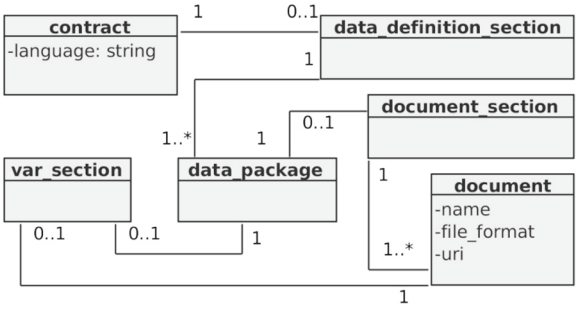

Figure 8 Ontological `data_definition_section` model.

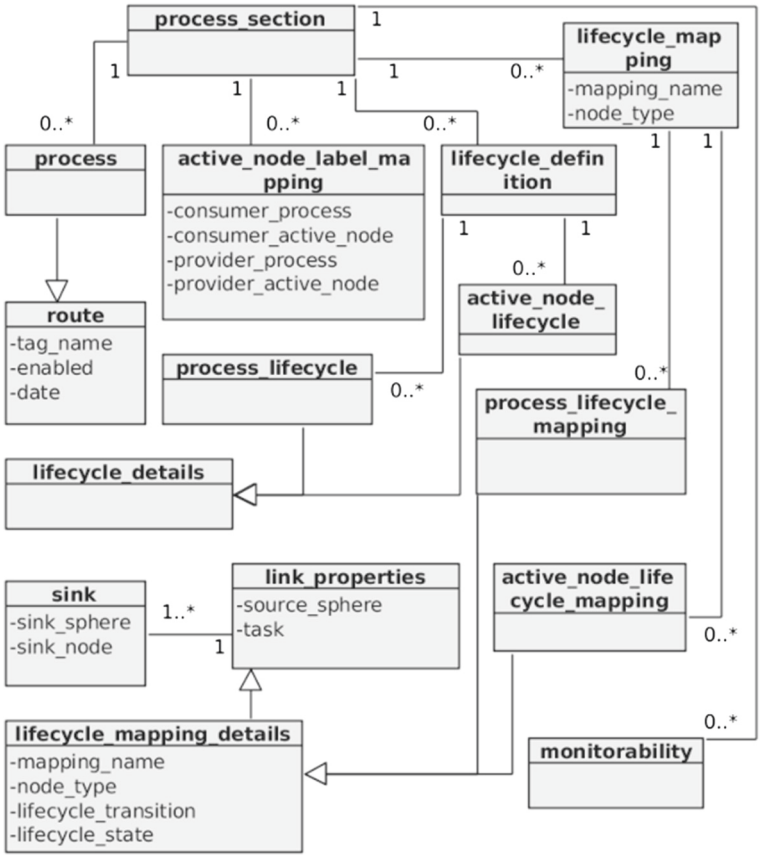

Figure 9 The `process` model.

instrumental for defining life-cycle stages of entire processes, or merely active nodes that are part of respective processes. Such definitions are part of consumer- and provider processes. The classes `process_lifecycle` and `active_node_lifecycle` are subclasses of `lifecycle_details`. The latter class is part of the eSourcing perspective that we explain in the sequel.

After specifying the lifecycles of respective processes and their contained active nodes, labels that express equal tasks have different expressions. Thus, class `active_node_label_mapping` is instrumental to define such semantic equivalence that is important for verifying projection inheritance of a consumer sphere and the refinement sphere of a service provider. Class `lifecycle_mapping` allows to map life-cycle stages of different processes and active nodes belonging to the domains of a service consumer and -provider. The mapping expresses such labels with different names are semantically equal.

3.2.3 Lifecycle-definition model

The model about different types of life-cycle elements in Figure 10 is an adjacent sub-model to `route` [25] with an associated class `lifecycle_details`. The classes of Figure 10 define the lifecycles of processes and active nodes that are part of an eSourcing configuration. Accordingly, Figure 10 depicts that `lifecycle_details` is a subclass of `lifecycle_elements`.

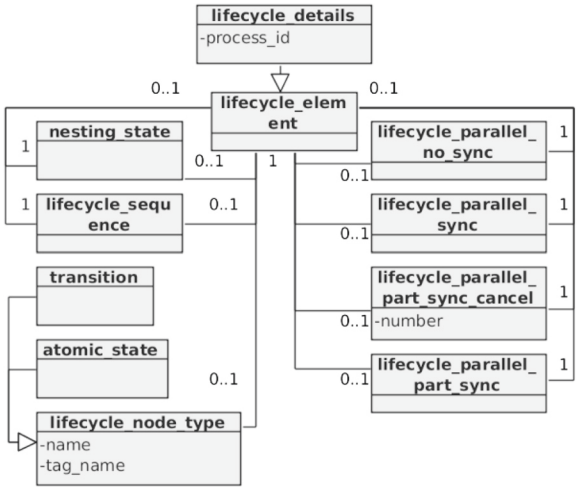

Figure 10 The `lifecycle_details` model.

There exists a mutual reference between class `lifecycle_elements` and the extracted lifecycle control-flow classes. Concretely, states are either of the type `nesting_ state`, or `atomic_state`. For the first class, on a lower level of a nesting state, further lifecycle elements exist, including another instance of `nesting_state`. On the other hand, class `atomic_state` is not further refinable on a lower level. Finally, transitions propel the lifecycle of a process, or active node from one state to the next.

3.2.4 Monitorability model

The classes in Figure 11 are part of the monitorability dimension of the eSourcing perspective to link active- and passive nodes that belong to the respective contractual spheres of a service consumer and -provider. On a process level, nodes are only active, i.e., task, transition, send task, receive task, send transition, receive transition, bi-directional task, and bi-directional transition. The only two cases of passive nodes exist on a life-cycle level of tasks where nested states and atomic states exist.

By defining monitoring links between nodes of respective eSourcing domains, it is possible for one contracting party to observe the progress of process enactment of the eSourcing counterpart. Usually the monitoring direction is from service consumer to provider. However, it is also possible that monitorability constructs of different directions are used in a P2P-collaboration.

The monitorability classes of Figure 11 fall into the categories polling-, or messaging constructs. In polling, the consumer frequently requests the status of the linked node in the domain of the service provider. Upon perceived enactment change, the linked node in the domain of the service consumer follows the change. Polling an active node with a life-cycle follows a mirroring of state changes, or transition firings. Polling a transition on a process level returns information about the firing of a linked node. The class `enactment_propagation` is for signalling the enactment of an eSourcing sphere starts in the domain

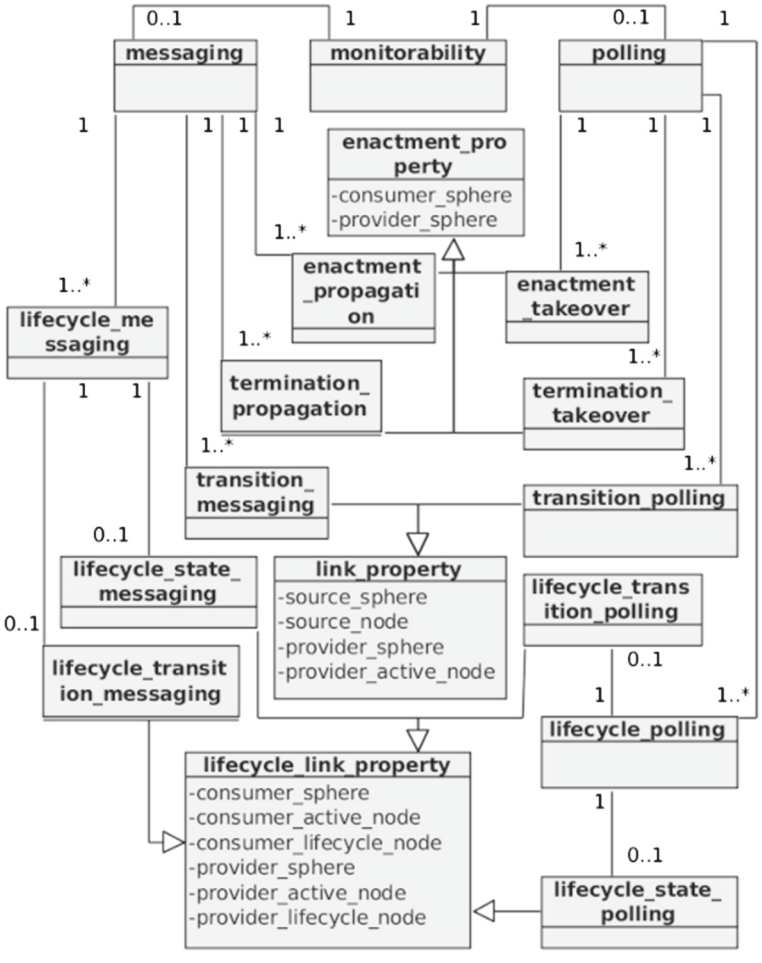

Figure 11 The `monitorability` model.

of the service consumer. This enactment commencement propagates to the service provider. Messaging classes link nodes from the process enacting domain to the observing counterparty. Thus, for linking two transitions in opposing eSourcing domains, the firing message triggers behaviour mirroring by the linked node. For linked tasks, lifecycle-state and -transition changes are events the task in the counterparty domain follows. Finally, class `enactment_termination` is for polling the completion of a service provision, which the service consumer mirrors so that the remainder of the consumer's in-house process commences with enactment.

3.2.5 Transition-type model

In Figure 12, class `transition_type` is central for all active nodes belonging to the control-flow perspective and the conjoinment dimension of the eSourcing perspective. We refer to [25] for full details. As Figure 12 shows, class `transition_type` is also a central connection to the workflow-data perspective. The class `lock_type` is initially set for a data package, which the `lock_change` tag changes. The `common_var_attributes` contains properties for all simple and complex variables.

The class named `data` in Figure 12 is central for the workflow-data perspective. An instance of `data` optionally references one or many data-packages that contain different variables and/or document definitions.

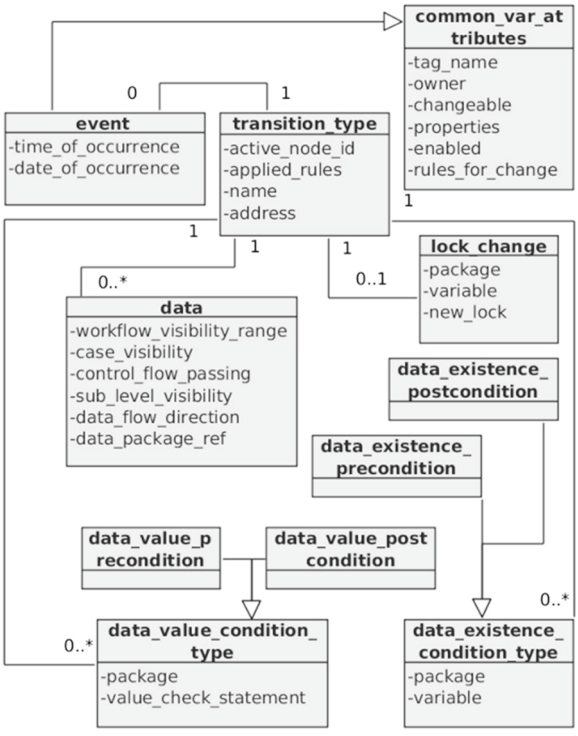

Figure 12 The `transition_type` model.

If `workflow_visibility_range` in `data` is true, a package is visible in all cases of a process template for all active nodes contained. If `case_visibility` is true, a data package is visible for all cases.

The `data_flow_direction` specifies the passing of data elements from a block-task instance to the corresponding sub-workflow that defines its implementation. When no further explicit assignment definition exists, no data passing happens since data has a global status. Thus, all lower-level elements are automatically aware of the data package. If additional assignment tags exist then edata passing uses either a dedicated data channel, or an integrated control- and data channel, i.e., in the latter case data flows along control flow.

The property `control_flow_passing` is instrumental for supporting `data_flow_direction`. Upon defining `data_types` on a block level, setting `control_flow_passing` to true implies the use of an integrated control- and data channel. As a result, data flows from one node to the next along control flow.

By using `sub_level_visibility` in combination with a `data_package_ref` definition for a control-flow block element, we specify to which lower-level degree the `data_package_ref` is visible. For example, if a block has 5 lower levels of routing elements and level 4 we define in a `sub_level_visibility` tag then elements located on the lowest level do not have visibility of the data package, i.e., the 5th level below the definition level of the specific data package.

In Figure 12, `transition_type` references other classes that are part of the workflow-data perspective. The reference `data_existence_precondition` is to check the presence of a variable as a prerequisite for the enactment of an active node and `data_existence_postcondition` defines the presence of a variable as a postcondition that must hold after the completed enactment of an active node. The superclass `data_existence_condition_type` comprises properties for defining which variable in a package must exist.

Finally, the classes `data_value_precondition` and `data_value_postcondition` define pre- and postconditions for the enactment of active nodes. However, differently to the case above where variable existence is the criteria, here the variables must have a particular value. Therefore, the superclass `data_value_condition_type` contains a property for checking the value of a variable.

3.2.6 Data model

The classes in Figure 13 support further data-flow models and belong exclusively to the data-flow perspective. The central class is `data` that is replicated in the routing

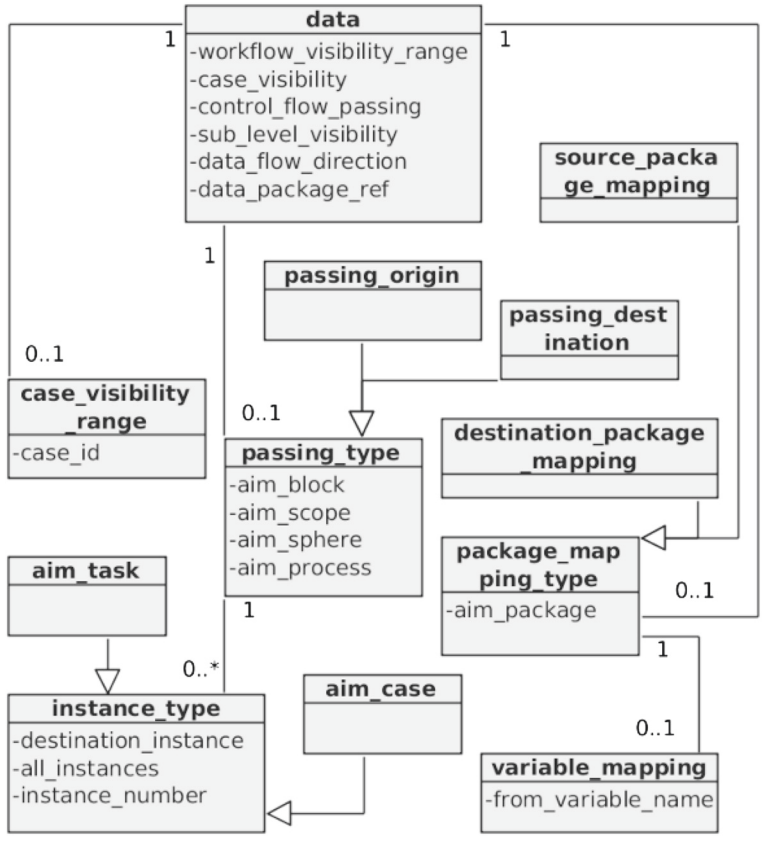

Figure 13 The `data` model.

models of [25] and Figure 12 to denote a connection between the respective models.

With `case_visibility_range`, we specify the set of cases where a data package is equally visible. Class `passing_destination` supports passing data packages between tasks. Class `passing_destination` is instrumental for two different scenarios. When `data_type` definition is on a block level, then explicit data passing from a block level to a contained lower-level element happens with specification `passing_destination`. Furthermore, a `passing_destination` specification assigns task-level data packages explicitly to the higher level, e.g., a block. Instead of passing a data package per se, `destination_package_mapping` and `source_package_mapping` map parts of a data package in one process node onto properties of another data package at some other location.

In Figure 13, `passing_type` is a superclass of `passing_origin` and `passing_destination`. Thus, depending on the visibility level for destination passing, the properties in `passing_type` are for setting detailed definitions in the subclasses `passing_origin` and `passing_destination`. Identifying the referenced classes `aim_task` and `aim_case` gives several options. For example, several instances of a task exist within a case, or an eSourcing configuration is equally instantiated several times. Thus, the superclass `instance_type` identifies data visibility for either all instances, or a subset of either tasks, or cases.

Finally, class `package_mapping_type` in Figure 13 serves as a superclass of `destination_package_mapping` and `source_package_mapping`. The superclass contains a property for targeting a data package for mapping variables onto another package. During that mapping, a transformation function is available that employs `package_mapping_type` references class `variable _mapping`. Next, we address semantic expressiveness assurance of electronic contracting.

4 Expressiveness exploration

The eSourcing example in Figure 2 uses labelled Petri nets [48,49]. The special type of Petri nets used for the conceptual levels of resourced, namely workflow nets (WF-nets) [50], has one unique passive input node and one unique passive output node. Furthermore, all other active and passive nodes in a WF-net contribute to its

processing. WF-nets carry the property of *soundness* [51,52], which informally states after the completion of a net, only one token must remain in the unique passive output node and all other passive nodes must be empty. WF-nets present an opportunity to verify the soundness before enactment of an overall process for ensuring a smooth enactment, e.g. with the powerful tool Woflan [53].

Starting with the domain of the service consumer in Figure 2, an in-house process shows on the conceptual level a WF-net. The in-house process contains a subnet termed a consumer sphere that is visualized with a grey ellipse. On the border of the consumer sphere, labelled passive nodes are interface places. Only one interface place is *i*-labelled and only one is *o*-labelled. The other interface places are either *in* or *out*-labelled to denote an exchange direction of business-critical information between the in-house process and its contained consumer sphere. Furthermore, the labelling implies whether an interface place has an input arc or an output arc in the sphere. If an interface place is *i*-, or *in*-labelled, it has one output arc to an active node in the sphere. If an interface place is *o*-, or *out*-labelled, it has one input arc from an active node in the sphere.

The in-house process is mapped to the internal level of Figure 2 onto legacy systems. A service provider enacts the consumer sphere and therefore projected to the external level to become the consumer contractual sphere. From the opposite eSourcing domain a, complementary provider contractual sphere is projected to the external level. Since the respective contractual spheres in Figure 2 are isomorph, a consensus is given between the eSourcing parties, which is the prerequisite for a contract [54].

The provider contractual sphere is complemented by a provider sphere on the conceptual level. Compared to the provider contractual sphere, additional nodes refine the provider sphere. In Figure 2, such refinement we depict by unlabelled active nodes in the provider sphere that do not exist in the provider contractual sphere. Hence, the refinement remains opaque for the collaborating counterpart. If the isomorph external-level processes are connected graphs, the refinement must be in accordance with *projection inheritance* [55] that is informally defined as follows. If it is not possible to distinguish the behaviours of processes x and y when executing arbitrary active nodes of x, but when only the effects of active nodes that are also present in y are considered, then x is a subclass of y. Thus, process x inherits the projection of the process definition y while process x conforms to the dynamic behaviour of its superclass by *hiding* active nodes new in x. Furthermore, such processes in an inheritance relation always have the same termination options. Note that Woflan [53] is also instrumental for verifying projection inheritance.

For relating the consumer sphere, the respective contractual spheres, and the provider sphere, the obligatory requirement of *well-directedness* of an eSourcing configuration must be fulfilled. This requirement focusses on the interface places of the spheres, which are part of exchange channels between spheres and the remaining in-house process. An eSourcing configuration is well-directed when the interface places of the consumer sphere, the respective contractual spheres of the service consumer and provider, and the provider sphere are equal in number and labelling.

An eSourcing configuration is formally mapped to so called bilateral workflow nets [7] so that a *collapsing* procedure is applicable for checking the correct termination on an interorganizational level. On the top right side of Figure 2, the in-house process and the provider sphere fulfill the well-directedness requirement. The bottom of Figure 2 shows the collapsed net with a removed consumer sphere replaced with the provider sphere in the in-house process. As a result, the collapsed net must be a sound WF-net. If the projections to the external level result in isomorph contractual spheres that are connected graphs, the collapsed net must be a subclass net of the consumer in-house process according to projection inheritance. In any case, the overall process resulting from the collapsing procedure of an eSourcing configuration must always terminate correctly, i.e., be a sound WF-net.

Next, we discuss the practical application of the eContracting approach.

5 Feasibility evaluation

For the feasibility study, we translate the concepts and properties of the HermiT-reasoner [43] verified eSourcing ontology into a machine-readable language eSML for which Section 1 comprises footnote hyperlinks to the ontology and the website with the schema definition.

For evaluating eSML, we consider a business case from the automobile industry. Briefly and related to Figure 1, in the automobile industry, OEMs have several tiers of suppliers that agree to deliver systems collaboratively. For example, the OEM assembles cars with systems like a cockpit, or an engine, etc. These systems are manufactured by Tier 1 that gets the components for those systems from a Tier 2 supplier. By applying eSourcing with specifying the inter-organizational collaboration with eSML, we facilitate the complex coordination effort between collaborating parties. In [31], the reader finds further details about the background of the industrial case study for this feasibility evaluation.

For the remainder, Section 5.1 shows the structure of eSML and also gives code examples that stem from a case study with industry. In Section 5.2, we explain an existing system architecture that enables the collaboration of decentralized autonomous organizations in a smart-contracting way that utilizes process views. Next,

Section 5.3 explains the lifecycle for setting up smart contracts based collaborations, including how to let them evolve. Finally, Section 5.4 show the lifecycle of eSML instantiations and enactment.

5.1 eSourcing markup language

We show the high-level structure of the business-collaboration language. As explained earlier, eSML uses parts of the ECML [44] schema as a foundation. Figure 14 reflects this fact by considering an entire eSML instance as a contract between collaborating parties and by structuring the eSML content into the blocks Who, Where, and What, as explained in Section 3. We refer to [25] for more information about ECML the definition of company data and company-contact data and the Where block.

The bold typed eSML-definitions in Figure 14 are extensions and modifications that are not part of the ECML foundation. In the Who block, extensions for eSML are the resource definition and the data definition. In the What block, the XRL adoption permits the use of control-flow patterns for business-process definitions that have semantic clarity. However, extensions exist for adopting the conjoinment nodes described in Section 3.2 and for linking to the resource- and data-definition sections of eSML that are both based on respective pattern collections [56,57]. The life-cycle definitions [25] are for the business processes and contained tasks.

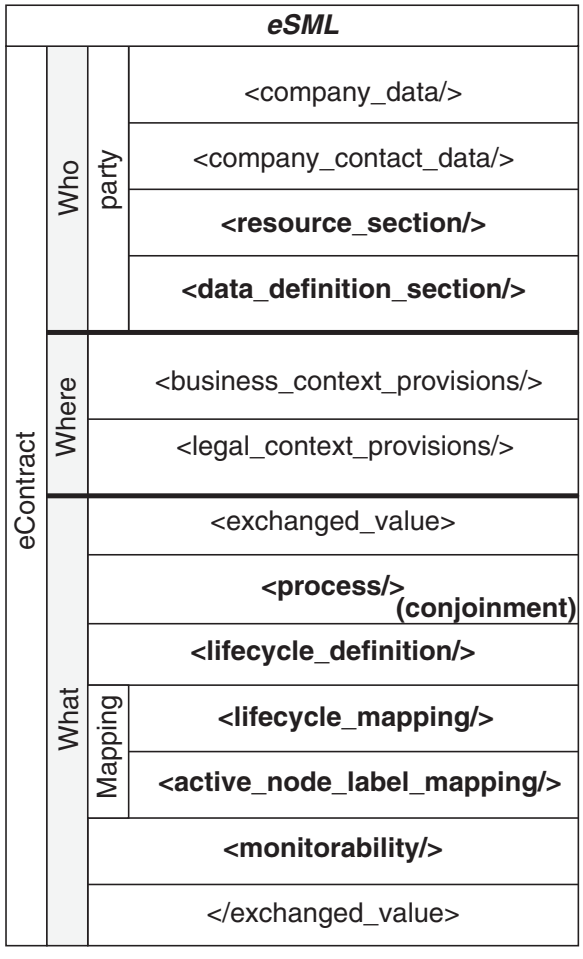

Figure 14 An overview of an eSML instantiation.

The life-cycle-mapping block addresses establishing semantic equivalence between, firstly, the life-cycles of the inter-organizationally harmonized business processes, and secondly, for the life-cycles of tasks from the opposing domains. Different labels of tasks belonging to processes of opposing domains may be semantically equal. To establish a semantic equality, the second part of the mapping block focuses on the mapping of task labels in the `active_node_label_mapping` tag. Such mapping is relevant for establishing a contractual consensus between collaborating parties. The monitorability (see Section 3.2) block of Figure 14 specifies how much of the enactment phase the service consumer perceives. Next, we show eSML examples that result from a CrossWork case study [31] and refer to [25] for the full eSML schema, models and more code examples.

5.1.1 Resource-perspective definition

The code extract in Listing 1 is part of the resource definition where an organizational unit is defined as a permanently existing organization. Thus, it is not a unit that dissolves at a certain point in time, e.g., an organization set up for the purpose of managing a project that has a deadline. In Line 11 the name of the organizational unit is defined, followed by the definition of the start date. Organizational units may have a business objective assigned. In the code example of Listing 1 this definition is omitted.

Listing 1 Resource-definition specification example in eSML.

```
10 <permanent_organizational_unit>
11  <name>Procurement_Department</name>
12  <start_date>2005-01-01</start_date>
13  <description/>
14  <business_objectives/>
15  <resource_nref>
16   <resource_type_ref>
17     Department_Head
18   </resource_type_ref>
19   <number>1</number>
20  </resource_nref>
21  <resource_nref>
22   <resource_type_ref>
23     Department_Clerk
24   </resource_type_ref>
25   <number>33</number>
26  </resource_nref>
27  <individual_resource>
28    Actor2
29  </individual_resource>
30 </permanent_organizational_unit>
```

The organizational unit has department members that are defined as roles. These roles are separately specified in the resource section of the eSML file. According to Line 19 the procurement department has one head of department. In Line 17, the `Department_Head` is a unique identifier for an separate extensive role definition. Similarly, Lines 22-24 specify that the procurement department has a resource assigned with the role name `Department_Clerk`. In the line below the number of individuals is given that slip into the mentioned role of a clerk. Finally, in Line 28 an individual is directly specified as a procurement-department member with the identifier `Actor2`.

5.1.2 Data-flow definition

The central part of the data-flow definition in an eSML file is a data package. Such data packages flow through a business process and are exchangeable to the opposing organizational domain. Following the data-flow pattern specifications [57], in an eSML instantiation, data has a particular visibility ranging from only a task to all instances of a business process and even its environment; a certain interaction type that focuses on the way data is communicated with, e.g, a data package is communicated from a task to another task; or from a block to a different process instance, and so on. Data has different transfer type specifications, e.g., by a copy, a reference, by value, etc. Finally, a data-flow element interacts with the control-flow perspective, e.g., a pre- or postcondition of data existence for a task, data-value based condition evaluation, etc.

Listing 2 Data-definition specification example in eSML.

```
10 <data_definition_section>
11   <data_package>
12    <package_id>cd</package_id>
13    <var_section>
14      <string_var
15        tag_name="Bill of Material"
16        var_id="BOM"
17        changeable="false"
18        enabled="enabled">
19          Surrounding Box; Gearing
20      </string_var>
21    </var_section>
22    <document_section>
23     <document>
24       <document_id>
25        cadDrawing
26     </document_id>
27      <name>
28       Cad Drawing of complete GearBox
29      </name>
30      <uri>
31        http://www.ve.com/gearBox.3ds
32       </uri>
33      </document>
34     </document_section>
35    </data_package>
36  </data_definition_section>
```

The code extract of Listing 2 specifies a data package with a contained variable and a document section. In Line 14, a variable is specified with its attributes. The bill of material is changeable, i.e., the value may be modified, and it is enabled for use. Additionally, Lines 24-32 define a document that is a CAD drawing and is available at a particular `uri`.

Listing 3 Data-package transfer example in eSML.

```
10 <receive_transition
11   active_node_id="CO"
12   name="Receive_Order">
13   <data>
14    <data_flow_direction>
15     input
16    </data_flow_direction>
17    <data_package_ref>
18      cd
19    </data_package_ref>
20   </data>
21   <data>
22    <data_flow_direction>
23      input
24    </data_flow_direction>
25    <data_package_ref>
26     do
27    </data_package_ref>
28   </data>
29 </receive_transition>
```

Listing 3 is an example of data packages are used with elements in the control-flow of an eSML instantiation. A receive node is specified that receives two data packages from the domain of a collaborating counterpart. In Line 13 the data package with the identifier *cd* is specified as an input. Additionally, in Line 17 the data package with the identifier *id* is equally input to the receiving node.

5.1.3 Structure of process-harmonization definition

The code extract in Listing 4 shows how to harmonize the structure of processes. For every collaborating party an `exchanged_value` section specifies if a service is either provided or consumed, which depends on the role a collaborating party slips into.

Listing 4 Business-process harmonization example comprising several collaborating parties in eSML.

```
10  <exchanged_value>
11      <service>
12      <process_section>
13        <process
14       tag_name="Gearbox_Production"
15       process_id="GB_production">
16       <parallel_sync>
17        <sourcing_sphere>
18          omitted control-flow routing elements
19        </sourcing_sphere>
20        <sourcing_sphere/>
21        <sourcing_sphere/>
22      <parallel_sync/>
23     </process>
24      <lifecycle_definitions/>
25      <lifecycle_mappings/>
26      <active_node_label_mapping/>
```

```
27      <monitorability/>
28      </process_section>
29     </service>
30    </exchanged_value>
```

Several `sourcing_spheres` may be part of a process specification. Lines 17-21 of Listing 4 specify that three sourcing spheres are embedded in a `parallel_sync` construct. Thus, the sourcing spheres are contained in parallel branches of control. The sourcing spheres are matched by spheres in `exchanged_value` sections of the same eSML instantiation that belong to opposing collaborating parties. For the matching, we specify grey-box contractual visibility pattern (see Section 3.2). In its final state where a consensus is specified in an eSML instantiation, the content of opposing sourcing spheres must match in content. In Lines 17-21, we omit control flow code for space limitation. In Lines 24-26, further eSML constructs specify further inter-organizational business process harmonization. The constructs for mapping life-cycles and for monitorability specifications are in the `exchanged_value` section of the service consumer.

5.1.4 Life-cycle definition

In an eSourcing configuration, the heterogeneous system environment of the internal level needs to be inter-organizationally harmonized. The business processes of collaborating parties may have deviating life-cycles on a process and task level. For the enactment phase, it is relevant to specify a synchronization of the life-cycles.

Listing 5 Lifecycle-definition example in eSML.

```
10 <lifecycle_definitions>
11   <process_lifecycle>
12   <lifecycle_sequence>
13    <atomic_state
14    name="VE_process_ready"
15    tag_name="ready"/>
16  <transition
17    name="VE_process_start_enactment"
18    tag_name="start_enactment"/>
19    ....more
20    tag_name="ended"/>
21     </lifecycle_sequence>
22    </process_lifecycle>
23    <active_node_lifecycle/>
24  </lifecycle_definitions>
```

In Listing 5, we shown that life-cycles for a process are specified with control-flow constructs. In Line 13, a not further decomposable atomic state is defined. However in a life-cycle, states are possible that contain further nested states. Accordingly, eSML contains a `nesting_state` construct that comprises lower-level states. The life-cycle of a process or a task is propelled by transitions of which Line 16 shows an example. In Line 23, the `active_node_life-cycle` of a task is defined with the same control-flow constructs for the specification of process life-cycles. We refer the reader to [25] for code examples about mapping lifecycle definitions between different tasks and processes belonging to separate counterparties.

5.1.5 Mapping definitions

If a heterogeneous system environment with different life-cycles is harmonized in one eSourcing configuration, it may be important for the enactment infrastructure to specify in an eSML instantiation how the respective life-cycles fit together. As the previous case study shows, in an eSourcing configuration several service providers are included with one service consumer. Thus, for life-cycle harmonization it is relevant to include all service providers. The respective life-cycle steps that are specified as equal may have diverting names but are still semantically equivalent. The same holds for the mapping of task labels from the domains of opposing parties.

The code extract below shows how life-cycles are mapped. In Line 11 the mapping of process life-cycles starts with first specifying the life-cycle label of the service consumer. From Line 17 onwards the semantically equivalent labels of two service providers are specified. For every life-cycle step this specification needs to be repeated.

Listing 6 Lifecycle-definition mapping example in eSML.

```
10 <lifecycle_mappings>
11   <process_lifecycle_mapping
12    mapping_name="process_ready"
13    node_type="lifecycle_state">
14    <consumer_sphere>
15     OEM_Sphere1
16    </consumer_sphere>
17    <consumer_active_node>
18     OEM_process_ready
19    </consumer_active_node>
20    <provider>
21    <provider_sphere>
22    Provider_SP1_1
23    </provider_sphere>
24    <provider_active_node>
25   SP1_process_idle
26    </provider_active_node>
27    <provider_sphere>
28    Provider_SP2
29    </provider_sphere>
30    <provider_active_node>
31    SP1_process_idle
32    </provider_active_node>
33    </provider>
34    </process_lifecycle_mapping>
35    <active_node_lifecycle_mapping
36    mapping_name="node_complete"
37    node_type="lifecycle_transition">
38    <consumer_sphere>
39    OEM_Sphere1
40    </consumer_sphere>
41    <consumer_active_node>
42    OEM_active_node_complete
43    </consumer_active_node>
44    <provider>
45    <provider_sphere>
46    Provider_SP1_1
47    </provider_sphere>
```

```
48    <provider_active_node>S
49    P1_active_node_complete
50     </provider_active_node>
51     <provider_sphere>
52     Provider_SP2
53     </provider_sphere>
54     <provider_active_node>
55     SP2_active_node_complete
56      </provider_active_node>
57     </provider>
58    </active_node_lifecycle_mapping>
59   </lifecycle_mappings>
```

For the mapping of task labels, a code extract is given in Listing 7. In Line 11 the specification of a service-consumer task label starts by first naming the process a task is contained in followed by the label of a task. In Line 17 similar specifications are given for the domain of the service provider. Differently to life-cycle mappings, there is always one label of the service consumer that is mapped to a label of one service provider because the concept of eSourcing assumes a task is always serviced by one provider.

Listing 7 Mapping active node labels in eSML.

```
10 <active_node_label_mapping>
11  <consumer_process>
12  GB_production
13  </consumer_process>
14  <consumer_active_node>
15  CO
16  </consumer_active_node>
17  <provider_process>
18  PP_SP1_1
19  </provider_process>
20  <provider_active_node>
21  Local_CO
22  </provider_active_node>
23 </active_node_label_mapping>
```

5.1.6 Monitorability definition

The code in Listing 8 is for specifying monitorability links with between the business processes of collaborating parties. The more monitorability patterns are specified, the more enactment progress the service consumer is able to follow. Many monitorability patterns are specified in [32] to cater for differing linking functionalities in a heterogeneous system environment of eSourcing configurations.

Listing 8 Monitorability specification in eSML.

```
10 <monitorability>
11  <polling/>
12  <messaging>
13  <transition_messaging>
14  <consumer_sphere>
15  SP1_Sphere1
16  </consumer_sphere>
17  <consumer_active_node>
18  CO
19  </consumer_active_node>
20  <provider>
21  <provider_sphere>
22  PP_SP1_1
23  </provider_sphere>
24  <provider_active_node>
25  Local_CO
26  </provider_active_node>
27  </provider>
28  </transition_messaging>
29  <messaging>
30 </monitorability>
```

An example for a monitorability specification is given in Listing 8 with two parts, namely one for the specification of polling constructs and one for specifying messaging constructs. In Lines 13-19, a transition-messaging monitorability construct is defined. First, the transition identifier in the domain of the service consumer represents the target node. In Lines 20-27, the source node for the messaging construct is specified located in the domain of the service provider.

5.2 eContracting architecture

To enable the setup and enactment of process-view based collaboration evolution, a system must meet a set of requirements. First, there must exist a service that facilitates the matching of service offers from collaborating parties and service requests from consuming organizations. Second, the collaborating parties house internally a component for the distributed binding and enactment of emergency cases. Third, with tool support, the parties must rapidly develop service offers and concrete services. Fourth, each collaborating party is capable of orchestrating its own internal legacy system for automating the collaboration. Finally, due to the heterogeneity of the collaboration, a translation service must exist for bridging the differences (technical, syntactic, semantic, pragmatic) between collaborating parties.

The eSourcing Reference Architecture [33] supports these requirements. Figure 15 depicts the resulting architecture in UML-component diagram notation that takes into account the above listed requirements. The *ServiceHUB* [42] as a trusted third party service in the middle that satisfies the first requirement and is suitable for the rapid setup phase in an emergency scenario. Each party has on an external layer an *eSourcing_Middleware* for the technical binding after a successful setup that satisfies the second requirement. During the distributed collaboration-enactment, the *eSourcing_Middleware* exchanges data via a security-ensuring gateway with the other parties. Thus, the *eSourcing_Middleware* also comprises external workflow- and rules-enactment services that coordinate each other not only internally but also via the gateway with other parties. We assume there exists in each party a conceptual layer with a service for *Setup_Support* that satisfies the third requirement and comprises tools for not only rapidly internally designing services and rules with the help of pattern libraries [58], but also includes a local verification- and simulation service. Next, each party has an internal layer with a service for *Legacy_Management*

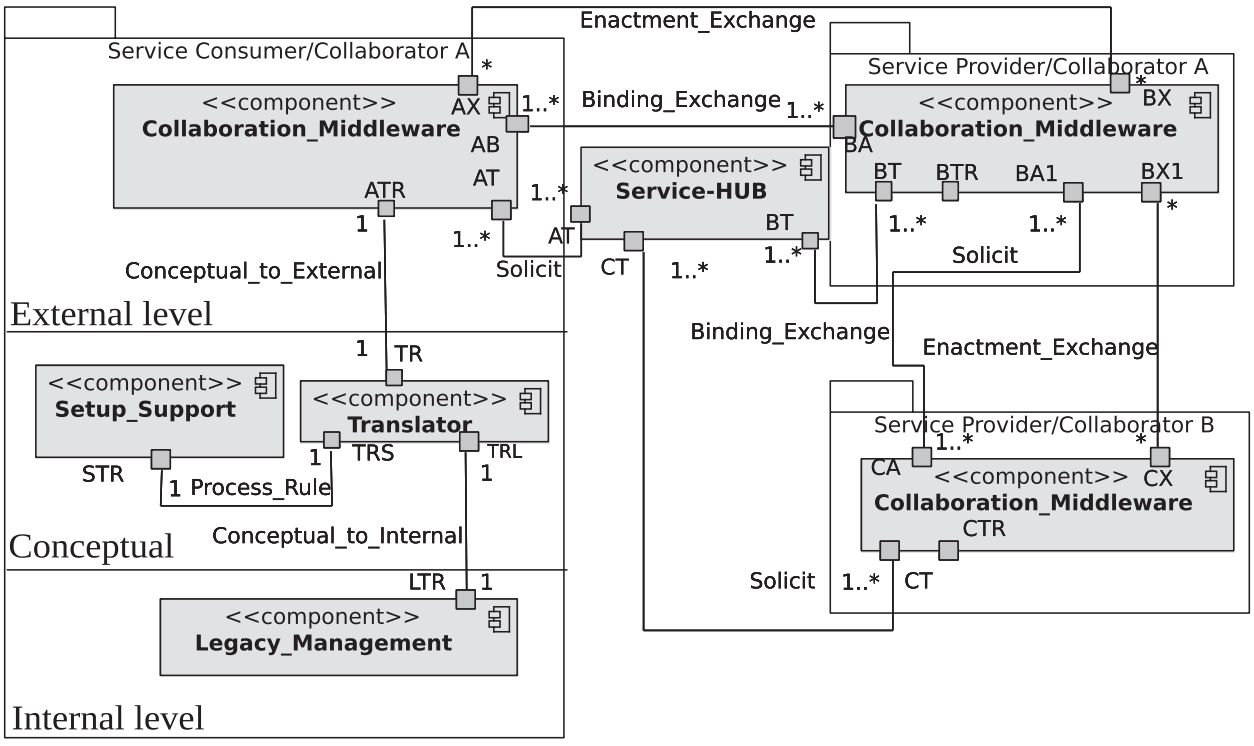

Figure 15 The eSourcing Reference Architecture eSRA [33] supports setting up eContracts.

that satisfies the fourth requirement and comprises local workflow- and rules enactment services that coordinate each other for the orchestration of Web-service wrapped internal legacy systems. Finally, the external- and internal enactment services exchange via a *Translator* service on the conceptual layer of each party to bridge the heterogeneous collaboration aspects. The *Translator* satisfies the final requirement and also connects on the conceptual layer with the *Setup_Support* service. Note that we omit in Figure 15 the conceptual- and internal layer with the exception of the *Hiring company* due to space limitations.

5.3 eContracting lifecycle

The generalized lifecycle in Figure 16 depicts conceptually and clockwise on the one hand the initial steps for setting up a cross-organizational collaboration configuration with the components of the system architecture in Figure 15. On the other hand, a subset of the lifecycle steps allows for a process-view based evolution of a collaboration configuration.

We first explain below the elementary steps of the lifecycle in Figure 16 relate to the architecture in Figure 15. Alphabetic letters relate the explanations to the respective lifecycle steps. Support for the conceptual formulation (a) of business processes and their accompanying rules involves the *eSourcing_Setup_Support* component. The latter comprises functionality for re-using business rules and process patterns for rapid conceptual-layer business process formulation.

Mapping details from the conceptual layer business-process to the internal layer (b) pertains to binding the tasks of a conceptual-layer process involving the *eSourcing_Setup_Support* to the Web-service ports that wrap legacy systems in the *Legacy_Management* so that an enactment-time orchestration of the latter is possible. The mapping also involves the *Translator* component from a conceptual- to an internal layer. Projecting from the conceptual layer business-process details to the external layer (c) involves the *Translator* component that creates different notation-formats to cater for business-process heterogeneity, e.g, from BPEL on the conceptual layer to BPMN on the external layer.

Brokering capability of projected business processes (d) for both the service consumer and the service provider must be able to place their projected process views into a broker environment of a *Service-HUB* component. This functionality is important for collaborations in an anonymous environment. The process views must be searchable for potential business partners. Bidding capability for projected processes views (e) is part of the *Service-HUB* component. The collaborating counter-party evaluates and chooses the subjectively best bid.

Negotiation support for setting up a collaboration configuration with known collaborating parties (f) is relevant

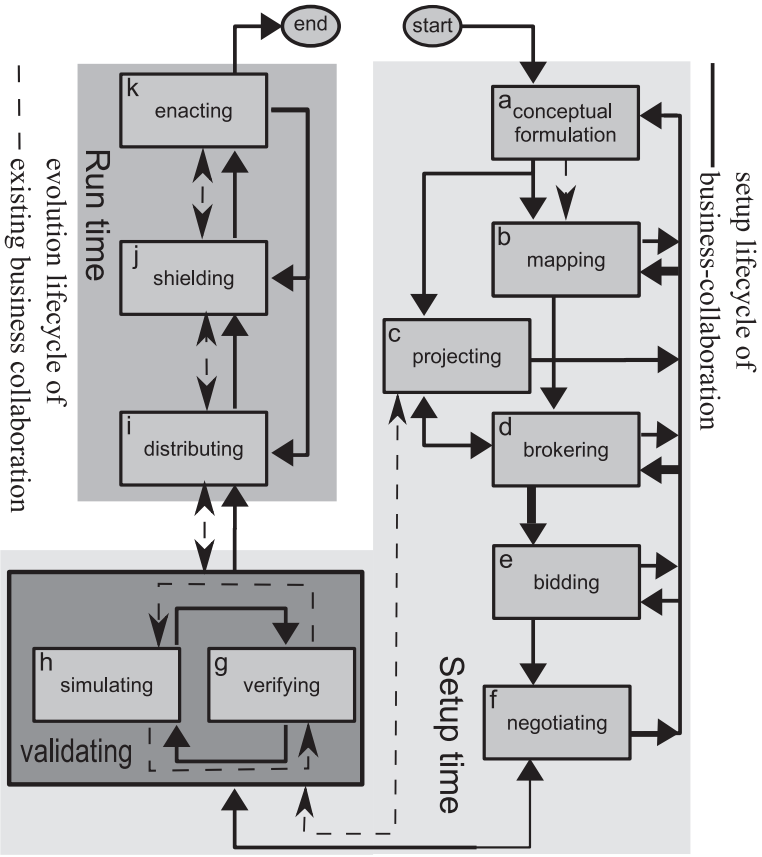

Figure 16 Lifecycle for setting up an eSourcing configuration (solid arrows) and evolving existing collaborations (dashed arrows) using process views.

after collaborating parties have found each other, they need a *Service-HUB* component for starting the contracting negotiations on the external layer of a collaboration configuration. This negotiation involves the projection of process views onto the external layer until achieving a matching that establishes a consensus between the service provisioning and service consumption.

Verifying perspectives of a collaboration configuration (g) from a control-flow point of view is important to verify a collaboration configuration for correct termination [7]. A verification must ensure that a service provisioning internally adheres to the externally promised collaboration behaviour. Simulation of a collaboration configuration (h) addresses that despite verification, errors may still occur during service enactment. Hence, a simulation component for business processes must be available for a-priori enactment simulation.

Distribution of business processes (i) to the external- and internal layer we cater for in the *Translator* component on the conceptual-layer. Shielding of business processes and legacy systems on concern-separating layers (j) must ensure the legacy systems that are Web-service wrapped and part of the internal layer, are safeguarded by data-monitoring functionality. Finally, the Enactment of a ready collaboration configuration (k) takes place with distributed rules- and process engines that are on the one hand part of the external layer's *eSourcing_Middleware* component and on the other hand, the *Legacy_Management* component of the internal layer.

After a completed setup phase, the enactment of a collaboration configuration commences. The actual enactment components must be present on an internal layer for orchestrating legacy systems. Additional enactment components on the external layer need to choreograph the internal components of the respective collaborating parties.

Finally, besides the full collaboration-setup lifecycle in Figure 16 that culminates in the enactment stage, there is a subset-lifecycle embedded denoted by dashed arrows for process-view based collaboration evolution. The start is from (k) to (a) for first performing changes to the existing internal business process. Next, the internal-process projection (c) to the external-layer process view that culminates in a verification (g) and also optional simulation (h). Changes to the process view must be propagated into the domains of collaborating counterparties. In the latter

case, a verification (g) and simulation (h) assures the re-established soundness of the overall business collaboration. Distribution (i) and shielding (j) precede a continued enactment (k) of the changed business collaboration. A dashed, bi-directional arrow in Figure 16 denotes that a faulty exception occurring at a specific lifecycle stage leads to a rollback into a previous lifecycle stage.

5.4 *eSML* enactment

As part of eSML we adopt for control-flow specifications the e*X*changable *R*outing *L*anguage XRL as an instance-based workflow language that uses XML for the representation of process definitions and Petri nets [48,49] for its semantics. The definition of XRL [59] contains as routing elements a catalog of control-flow patterns [60-63] that result in strong control-flow expressiveness. These routing elements are equipped with Petri-net semantics [64], namely, every routing element stands for an equivalent *workflow net* (WF-net) [50,51,65] that can be connected with other routing elements into a bigger WF-net.

An XRL route is a consistent XML document, that is, a well-formed and valid XML file with top element route [25]. The structure of any XML document forms a tree. In case of XRL, the root element of that tree is the route that contains exactly one so-called routing element. A routing element is an important building block and can either be simple (no child routing elements) or complex (one or more child routing elements). A complex routing element specifies whether, when and in which order the child routing elements are carried out.

To evaluate the expressiveness of eSML, the control-flow specification realizes the WF-net semantics of XRL by mapping to PNML [66-68], an XML-based interchange format that permits the definition of Petri-net types. A style-sheet translator contains mapping rules [64] to PNML for every XRL control-flow construct.

Due to page limitation, in Figure 17, we can only explain the lifecycle of a business process as it is carried out by the enactment application XRL/flower [69] that is adoptable for an eSRA-based implementation. Woflan [51,70] for checking control-flow soundness, is part of XRL/flower. Note that new control-flow elements adopted in XRL merely require an additional mapping rule in the stylesheet translator while the enactment engine remains unchanged. We refer to [25] for further details.

6 Related work

Contracting is part of Web-service choreography in some research work that only takes a technical position. In [71], contracts are descriptions of the observable behaviour of multiple services to tackle the problem of composition as sets of inout- and output actions. The authors show that a compliant group of contracts is still compliant after replacement by one of its subcontract. In [72], the same authors relate the theory of contracts with the notion of choreography conformance, used to check whether an aggregation of services correctly behaves according to a high level specification of their possible conversations based on input- and output actions. Projection and contract refinement achieve composition of choreography.

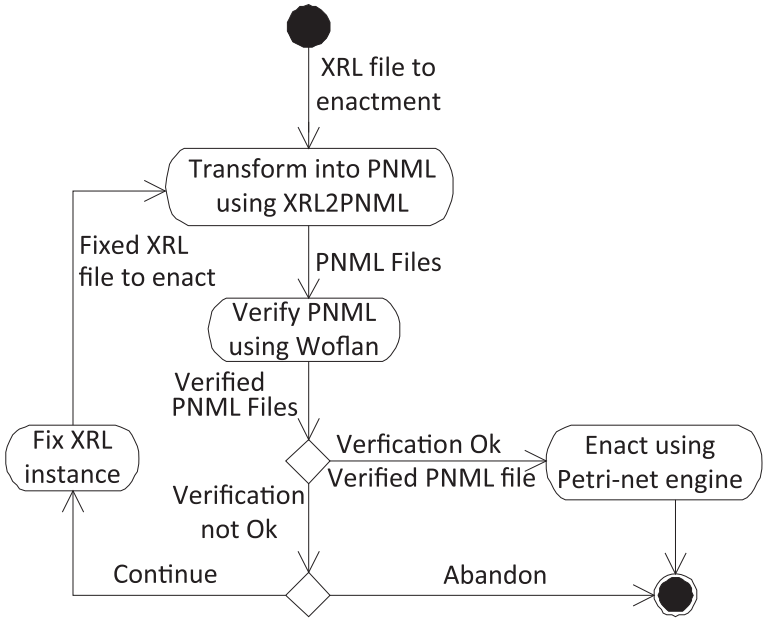

Figure 17 Lifecycle of eSML-business-process instantiation and enactment.

The authors in [73] state support for automated service contracting and enactment is crucial for any large scale service environment, where large numbers of clients and service providers interact. Concurrent Transaction Logic is instrumental to model and reason about service contracts to allow iterative processes in the specification of service contracts and enable reasoning about such contracts. Thereby, the authors limit themselves in their notion of contracts to technical service choreographies, service policies and contract requirements.

In [74], the authors consider contracts as labelled transition systems over located action names, representing operations at a certain location over a network. In this technicality focussed approach, the authors study the foundational aspects of contract compliance in a language independent way. The language independent representation of contracts allows for choreography projection in structured operational semantics. The evaluation applies the theory of contract compliance with industry-standard choreography specifications such as WS-CDL.

The related work has in common that no sociotechnical approach for the concept of contracting in service choreography recognizes the interaction between people and technology in cross-organizational collaborations. We address this gap by choosing a reality-based notion of a legal contract that states a consensus between collaborating parties must be present. This consensus in eSML represents the matching of process views by a service consumer and service provider, which is fundamentally different to the listed related work of purely technical focus.

Various aspects of negotiating and agreeing contracts between software agents acting on behalf of enterprises or individuals are described in [75].

7 Conclusions

This paper presents the ontological concepts and properties of smart contracting that is an essential ingredient for the management of decentralized autonomous organizations. The resulting eSourcing ontology that we define in the ontology language OWL and check with the HermiT reasoner, is input for developing the eSourcing Markup Language eSML. The latter is a choreography language for cross-organizational business collaboration. eSML results from a sociotechnical methodical, case study-based suitability and expressiveness exploration that ensures the language comprises essential collaboration concepts with a foundation for semantic clarity.

As eSML adopts a real-life contracting foundation, collaborating parties use process views they project externally for cross-organizational matching. Once a matching occurs, a consensus exists that is the essential criteria for establishing a contract. The process views are subsets of larger business processes inside the domains of collaborating parties. We use a subset of ECML as a base language for eSML that we enhance with additional schemas for process-view matching, and cross-organizational conjoinment and monitorability for achieving suitability. Furthermore, assuming that the control-flow perspective in a business collaboration is best explored, the expressiveness in eSML we address by adopting WF-net based semantics that is verifiable with tool support. Employing setup-interaction patterns of collaborating parties, we evaluate eSML in a proof-of-construction prototype for the setup and enactment of business collaborations in the CrossWork research project.

For future work, we plan to further enhance the eSourcing ontology with law researchers towards providing a mature ontology for advancing smart contracting. Furthermore, we plan to carry out more case studies with eSML in research projects about designing cyber-physical systems with smart-object orchestration. In those studies, we want to address expressiveness extensions into more Internet of Things perspectives. Additionally, an important extension for the eSourcing ontology and eSML is an adoption of concepts and properties to cater for adopting advanced security assurance measures that are relevant for open cyber-physical systems collaborations. Another open research issue is the safeguarding of business collaborations with transactionality concepts that need to go further than traditional transactions from the database- and workflow domains, i.e., pertaining to blockchain-related variants such as sidechains, treechains, minichains, and so on. Thus, future work will explore such electronic business transactions with the objective of understanding how to extend eSML for ensuring a safeguarding of cyber-physical system collaborations, preferably by using blockchain technology.

Endnotes

[a] Principles of European Contract Law developed by the Lando-Comission in Europe http://www.jus.uio.no/lm/eu.contract.principles.parts.1.to.3.2002/

[b] UNIDROIT Principles of International Commercial Contracts 2010 http://www.unidroit.org/english/principles/contracts/principles2010/integralversionprinciples2010-e.pdf

[c] Principles, Definitions and Model Rules of European Private Law known as Draft Common Frame of Reference http://ec.europa.eu/justice/contract/files/european-private-law_en.pdf

[d] United Nations Convention on Contracts for the International Sale of Goods, adopted 11 April 1980.

[e] International Chamber of Commerce rules http://www.iccwbo.org/products-and-services/trade-facilitation/incoterms-2010/the-incoterms-rules/.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
AN is the leading author who invented the eSourcing collaboration framework based on which we deduce the eContracting concept of this paper together with the evaluation that cilminates in eSML. This claim is backed by the listed literature references where Alex Norta is an author. LM was involved in the earlier OrChor'14 workshop-paper version that lead to the invitation for the special edition of the Journal of Internet Services and Applications. LM performed extensive proof reading and editing the submitted version. YD was also involved in the earlier OrChor'14 workshop-paper version that lead to the invitation for the special edition of the Journal of Internet Services and Applications. He additionally redesigned the conceptual-property depictions of Section 3 that are input for the eSML schema definitions. AR and MK are law-scientists and checked the approach from a professional legal perspective. The findings entered the introduction of this paper for justifying the sound legal foundation of the work in this paper. KT checked the paper from an artificial agent perspective at the end of Section 3.1. All authors read and approved the final manuscript.

Acknowledgement
This paper was supported in part by CNSF grant 61363007 and by HNU grant KYQD1242 and HDSF201310; the Dawn Program of Shanghai Education Commission (11SG44); the Research Fund for the Doctoral Program of Higher Education of China (20123120130001).
The work is supported by the National Natural Science Foundation of China(NSFC) under Grant No. 61202376, Shanghai Leading Academic Discipline Project under Grant No. XTKX2012, and Shanghai Engineering Research Center Project under Grant No. GCZX14014 and C14001.

Author details
[1]Tallinn University of Technology, Akadeemia Tee 15A, 12618 Tallinn, Estonia. [2]Hainan University, Haikou, China. [3]University of Shanghai for Science and Technology, Shanghai, China.

Received: 30 October 2014 Accepted: 4 March 2015
Published online: 24 April 2015

References

1. Allen P, Higgins S, McRae P, Schlamann, H (eds.): (2006) Service Orientation: Winning Strategies and Best Practices. Cambridge University Press
2. Antonopoulos N, Gillam L (2010) Cloud Computing: Principles, Systems and Applications. Springer
3. Alonso G, Casati F, Kuno H, Machiraju V (2004) Web Services: Concepts, Architectures and Applications. Springer-Verlag, Berlin Heidelberg
4. Eshuis R, Norta A, Kopp O, Pitkanen E (2013) Service outsourcing with process views. IEEE Trans Serv Comput 99(PrePrints):1. doi:10.1109/TSC.2013.51
5. Jung JY, Kim H, Kang SH (2006) Standards-based approaches to b2b workflow integration. Comput Ind Eng 51(2):321–334. doi:10.1016/j.cie.2006.02.011
6. Khalaf R (2007) From rosettanet pips to bpel processes: A three level approach for business protocols. Data Knowl Eng 61(1):23–38. doi:10.1016/j.datak.2006.04.006
7. Norta A, Eshuis R (2010) Specification and verification of harmonized business-process collaborations. Inform Syst Front 12:457–479. doi:10.1007/s10796-009-9164-1
8. Butterin V (2014) A next-generation smart contract and decentralized application platform. White Paper
9. Szabo N (1997) Formalizing and securing relationships on public networks. First Monday 2(9). http://firstmonday.org/ojs/index.php/fm/article/view/548/469
10. Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system. Consulted 1(2012):28
11. Patron T The Bitcoin Revolution: An Internet of Money. Travis Patron
12. Swan M (2015) Blockchain Thinking: The Brain as a DAC (Decentralized Autonomous Organization. Texas Bitcoin Conference
13. Panikkar BS, Nair S, Brody P, Pureswaran V (2014) ADEPT: An IoT Practitioner Perspective. IBM
14. Lohmann N, Kleine J (2008) Fully-automatic Translation of Open Workflow Net Models into Simple Abstract BPEL Processes. In: Modellierung, vol. 12. Gesellschaft für Informatik E.V., Berlin, Germany. p 14
15. Decker G, Kopp O, Leymann F, Weske M (2007) Bpel4chor: Extending bpel for modeling choreographies. In: Web Services, 2007. ICWS 2007. IEEE International Conference On. IEEE. pp 296–303
16. Jordan D, Evdemon J, Alves A, Arkin A (2014) Web Services Choreography Description Language 1.0. http://www.w3.org/TR/ws-cdl-10/
17. Decker G, Barros A (2008) Interaction modeling using bpmn. In: BPM'07: Proceedings of the 2007 International Conference on Business Process Management. Springer, Berlin, Heidelberg. pp 208–219
18. Model BP (2011) Notation (bpmn) version 2.0. Object Management Group specification. http://www.bpmn.org
19. Zaha JM, Barros A, Dumas M, ter Hofstede AHM (2006) Let's dance: A language for service behavior modeling. In: Meersman R, Tari Z (eds). On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, and ODBASE. Lecture Notes in Computer Science, vol. 4276. LNCS Springer, Montpellier, France. pp 145–162
20. Eisenberg B, Nickull D (2001) ebXML Technical Architecture Specification, Technical report, Organization for the Advancement of Structured Information Standards (OASIS)
21. Motal T, Zapletal M, Werthner H (2009) The Business Choreography Language (BCL) - A Domain-Specific Language for Global Choreographies. In: SERVICES-2 '09: Proceedings of the 2009 World Conference on Services - II. IEEE Computer Society, Washington, DC, USA. pp 150–159. doi:10.1109/SERVICES-2.2009.25
22. Norta A (2015) eSourcing.owl ontology. Tallinn University of Technology. http://tinyurl.com/nogogon
23. Knublauch H, Fergerson RW, Noy NF, Musen MA (2004) The protégé owl plugin: An open development environment for semantic web applications. In: The Semantic Web–ISWC 2004. Springer. pp 229–243
24. Horrocks I, Motik B, Wang Z (2012) The hermit owl reasoner. In: OWL Reasoner Evaluation Workshop (ORE 2012). CEUR Workshop Proceedings, CEUR-WS.org
25. Norta A (2007) Exploring Dynamic Inter-Organizational Business Process Collaboration. PhD thesis, Technology University Eindhoven, Department of Information Systems
26. Nimmer RT (2007) The legal landscape of e-commerce: Redefining contract law in an information era. J Contract Law 23(1):10–31
27. Gaillard E (1995) Thirty years of lex mercatoria: Towards the selective application of transnational rules. ICSID Rev 10(2):208–231
28. Hevner AR, Ram S, March ST, Park J (2004) Design science in information system research. MIS Q 28(1):75–105
29. March ST, Storey VC (2008) Design science in the information systems discipline: an introduction to the special issue on design science research. Manag Inform Syst Q 32(4):6
30. Grefen P, Eshuis R, Mehandjiev N, Kouvas G, Weichhart G (2009) Internet-based support for process-oriented instant virtual enterprises. IEEE Internet Comput 13(6):65–73
31. Mehandjiev N, Grefen P (2010) Dynamic Business Process Formation for Instant Virtual Enterprises. Springer
32. Norta A, Grefen P (2007) Discovering Patterns for Inter-Organizational Business Collaboration. Int J Cooperative Inform Syst (IJCIS) 16:507–544. doi:10.1142/S0218843007001664
33. Norta A, Grefen P, Narendra NC (2014) A reference architecture for managing dynamic inter-organizational business processes. Data Knowl Eng 91(0):52–89
34. Alonso G, Fiedler S, Hagen C, Lazcano A, Schuldt H, Weiler N (1999) WISE: business to business e-commerce. In: Proc. of the 9th International Workshop on Research Issues on Data Engineering, Sydney, Australia. pp 132–139
35. Hoffner Y, Ludwig H, Gülcü C, Grefen P (2005) Architecture for Cross-Organizational Business Processes. In: Procs. 2^{nd} Int. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, Milpitas, CA, USA. pp 2–11
36. Lazcano A, Schuldt H, Alonso G, Schek HJ (2001) Wise: Process based e-commerce. IEEE Data Eng Bull 24(1):46–51
37. Kutvonen L, Ruokolainen T, Metso J (2007) Interoperability middleware for federated business services in web-Pilarcos. Int J Enterprise Inform Syst Spec Issue Interoperability Enterprise Syst Appl 3(1):1–21

38. Kutvonen L, Norta A, Ruohomaa S (2012) Inter-enterprise business transaction management in open service ecosystems. In: Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International. IEEE. pp 31–40
39. SAP (2015) Sourcing & Procurement Solutions. SAP.com. http://www.sap.com/solution/lob/procurement/software/sourcing/index.html
40. ORACLE (2015) ORACLE Sourcing. ORACLE.com. http://www.oracle.com/us/products/applications/ebusiness/procurement/053985.html
41. Ruokolainen T, Ruohomaa S, Kutvonen L (2011) Solving service ecosystem governance. In: Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International. IEEE. pp 18–25
42. Norta A, Kutvonen L (2012) A cloud hub for brokering business processes as a service: A "rendezvous" platform that supports semi-automated background checked partner discovery for cross-enterprise collaboration. In: SRII Global Conference (SRII), 2012 Annual. pp 293–302
43. Shearer R, Motik B, Horrocks I (2008) Hermit: A highly-efficient owl reasoner. In: CEUR Workshop Proceedings, Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, vol. 432
44. Angelov S (2006) Foundations of B2B Electronic Contracting. Dissertation, Technology University Eindhoven, Faculty of Technology Management, Information Systems Department
45. Reinecke JA (1984) Introduction to Business: A Contemporary View. Allyn & Bacon
46. Duan Y (2012) A survey on service contract. In: Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference On. IEEE. pp 805–810
47. Sterling L, Taveter K (2009) The Art of Agent-oriented Modeling. MIT Press
48. (1998) Lectures on Petri Nets I: Basic Models(Reisig W, Rozenberg G, eds.), Vol. 1491. Springer, Heidelberg, Germany
49. Lectures on Petri Nets II: Applications (1998)(Reisig W, Rozenberg G, eds.), Vol. 1492. Springer, Heidelberg, Germany
50. Ellis C, Nutt G (1993) Modelling and Enactment of Workflow Systems. In: Marsan M. A. (ed). Application and Theory of Petri Nets 1993, vol. 691. Springer, Heidelberg, Germany. pp 1–16
51. Aalst W (1998) The Application of Petri Nets to Workflow Management. J Circuits Syst Comput 8(1):21–66
52. Kindler E, Aalst W (1999) Liveness, Fairness, and Recurrence. Inform Process Lett 70(6):269–274
53. Verbeek H, Basten T, Aalst W (2001) Diagnosing Workflow Processes using Woflan. Comput J 44(4):246–279
54. Angelov S, Grefen P (2003) The 4W framework for B2B e-contracting. Int J Netw Virtual Organizations 2(1):78–97
55. Aalst W (2000) Inheritance of Interorganizational Workflows: How to Agree to Disagree Without Loosing Control?. BETA Working Paper Series, WP 46, Eindhoven University of Technology, Eindhoven
56. Russell N, van der Aalst WM, ter Hofstede AH, Edmond D (2005) Workflow resource patterns: Identification, representation and tool support. In: Advanced Information Systems Engineering. Springer. pp 216–232
57. Russell N, Ter Hofstede AH, Edmond D, van der Aalst WM (2005) Workflow data patterns: Identification, representation and tool support. In: Conceptual Modeling–ER 2005. Springer. pp 353–368
58. Norta A, Hendrix M, Grefen P (2006) On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, and ODBASE. In: Meersman R, Tari Z. (eds). Lecture Notes in Computer Science, vol. 4277. LNCS Springer, Montpellier, France. pp 834–843
59. Aalst W, Kumar A (2003) XML Based Schema Definition for Support of Inter-organizational Workflow. Inform Syst Res 14(1):23–47
60. Workflow Patterns Home Page. http://workflowpatterns.com/
61. Aalst W, Hofstede A, Kiepuszewski B, Barros AP (2000) Advanced Workflow Patterns. In: Etzion O, Scheuermann P (eds). 7th International Conference on Cooperative Information Systems (CoopIS 2000) Vol. 1901. pp 18–29
62. Kiepuszewski B (2002) Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows PhD thesis, Queensland University of Technology, Queensland University of Technology, Brisbane, Australia. citeseer.nj.nec.com/kiepuszewski02expressiveness.html
63. Kiepuszewski B, Hofstede A, Aalst W (2003) Fundamentals of Control Flow in Workflows. Acta Informatica 39(3):143–209
64. Aalst W, Verbeek HMW, Kumar A (2001) XRL/Woflan: Verification of an XML/Petri-net based language for inter-organizational workflows (Best paper award). In: Altinkemer K, Chari K. (eds). Proceedings of the 6th Informs Conference on Information Systems and Technology (CIST-2001). Informs, Linthicum, MD. pp 30–45
65. Aalst W (1997) Verification of Workflow Nets. In: Azéma P, Balbo G (eds). Application and Theory of Petri Nets 1997. Springer, Heidelberg, Germany Vol. 1248. pp 407–426
66. Billington J, Christense S, Van Hee K, Kindler E, Kummer O, Petrucci L, Post R (2003) The Petri Net Markup Language: Concepts, Technology, and Tools. In: Aalst W, Best E (eds). Proc. of the 24th International Conference, ICATPN 2003, Lecture Notes in Computer Science. Springer, Eindhoven, The Netherlands. pp 483–505
67. LIP6 (2015) Welcome to PNML.org. http://www.pnml.org/
68. Weber M, Kindler E (2003) The petri net markup language. In: Ehrig H, Reisig W, Rozenberg G, Weber H (eds). Petri Net Technology for Communication-Based Systems Advances in Petri Nets. Lecture Notes in Computer Science. Springer. p 455
69. Norta A (2004) Web supported enactment of petri-net based workflows with XRL/Flower. In: Cortadella J, Reisig W (eds). Proc. of the 25th International Conference on the Application and Theory of Petri Nets 2004. Lecture Notes in Computer Science. Springer, Bologna, Italy. pp 494–503
70. Verbeek H, Basten T, Aalst W (2001) Diagnosing Workflow Processes Using Woflan. Comput J Br Comput Soc 44(4):246–279
71. Bravetti M, Zavattaro G (2007) Contract based multi-party service composition. In: Arbab F, Sirjani M (eds). International Symposium on Fundamentals of Software Engineering. Lecture Notes in Computer Science, vol. 4767. Springer. pp 207–222. doi:10.1007/978-3-540-75698-9_14. http://dx.doi.org/10.1007/978-3-540-75698-9_14
72. Bravetti M, Zavattaro G (2007) Towards a unifying theory for choreography conformance and contract compliance. In: Lumpe M, Vanderperren W (eds). Software Composition. Lecture Notes in Computer Science, vol. 4829. Springer. pp 34–50. doi:10.1007/978-3-540-77351-1_4. http://dx.doi.org/10.1007/978-3-540-77351-1_4
73. Roman D, Kifer M (2008) Semantic web service choreography: Contracting and enactment. In: The Semantic Web-ISWC 2008. Springer. pp 550–566
74. Bravetti M, Zavattaro G (2009) Contract compliance and choreography conformance in the presence of message queues. In: Bruni R, Wolf K. n. (eds). Web Services and Formal Methods, Lecture Notes in Computer Science, vol. 5387. Springer. pp 37–54. doi:10.1007/978-3-642-01364-5_3. http://dx.doi.org/10.1007/978-3-642-01364-5_3
75. Ossowski S (2013) Agreement Technologies:Law, Governance and Technology. Springer