ISR
Institute for Software Research
University of California, Irvine

A Multidimensional Evaluation of Integrative E-commerce Architectures

Alegria Baquero
University of California, Irvine
abaquero@uci.edu

Richard N. Taylor
University of California, Irvine
taylor@ics.uci.edu

September 2012
ISR Technical Report # UCI-ISR-12-9

Institute for Software Research
ICS2 221
University of California, Irvine
Irvine, CA 92697-3455
www.isr.uci.edu

A Multidimensional Evaluation of Integrative E-commerce Architectures

ISR Technical Report #UCI-ISR-12-9
September 2012

Alegria Baquero and Richard Taylor

Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425
[abaquero|taylor]@ics.uci.edu

Abstract. The proliferation of goods and services offered online, as well as the growing number of e-consumers are catalysts for the ongoing burgeoning of e-commerce among Internet activities. Numerous industries have adopted e-commerce technologies to optimize and automate business processes. In this paper, we are mainly concerned on the synergic relationship between three fundamental components of e-commerce—negotiation, contracts, and business workflow. We conduct a survey on e-commerce technologies and evaluate this body of work through a multidimensional framework. On one dimension, our framework assesses the extent to which negotiation, contracts, and business workflow are integrated and interoperate within e-commerce architectures. On the other dimensions, we assess if and how desirable system properties such as decentralization, dynamic adaptation, automation, and security are supported within each of these e-commerce phases. Our findings show that despite the existence of studies approaching these concerns, full exploitation of co-dependent e-commerce components—negotiation, contracts, and workflow—is elusive. In addition, there is insufficient support for dynamic adaptation, automation, security, accountability, and bidirectional relationship among e-commerce components.

Keywords: Software engineering; Software architecture; Electronic commerce; Contracts; Negotiation; Workflow; Business Processes

Table of Contents

1 Introduction

Purchasing a multiplicity of products—such as books, clothes, and even groceries—as well as an array of services—such as digital music, airline tickets, and online movies—is today a matter of having an Internet connection and making a few clicks. The carefree consumer needs not to concern about the complex machinery of inter-organization processes set in motion. Behind user interfaces, market analysis, negotiation, procurement, production, alliances, and other interactions among industries occur. E-commerce has come a long way since its beginnings in the mid-1990's, now comprising a leading activity within the Internet.

Business-to-business (B2B) e-commerce technologies have spread through many industries in the effort to optimize and secure both intra- and inter-organizational business activities—supply chain management and information sharing, among others—and in so doing reduce administrative and operational costs, and increase trust between business partners. These technologies face the challenge of achieving system integration and interoperability, great aspirations within the e-commerce domain.

Integration and interoperation refer to the capability of using software components as building blocks to larger applications, which communicate and collaborate with each other. Integration and interoperation is an inherently difficult problem due to the heterogeneity of components and systems, their geographical distribution, the insecurity and unreliability of networks, the potential unavailability of required services, the divergent protocols and data formats, the differences in semantics and business ontologies, and so on. Integration and interoperation development efforts account for big costs within software projects including e-commerce applications, a challenging problem that comes at a high price, not only among organizations, but is a well-known in-house problem as well. Despite the inherent difficulty, the benefit of achieving such integration is the ability to build applications from heterogeneous components, communicate, collaborate, and do business with other organizations.

We are mainly interested in the integration and interoperation of the fundamental components or phases of the *e-commerce contracting lifecycle* (figure 1). This model describes the relationship and the kind of interactions taking place between search, negotiation, contract, and workflow.

A business relationship begins with an information phase where consumers — individuals or organizations — search for potential suppliers, request for quotes, and make initial offers. This activity is bypassed when the consumer has already chosen a supplier based on previous business relationships. Subsequently, parties negotiate the terms of the agreement and collaboratively modify a contract draft. When parties have reached an agreement, an electronic contract is established, describing permissions and prohibitions bestowed to each party. The contract guides individual and collaborative business activities—namely intra- and inter- organizational business workflows—that parties carry out to comply with the contract. When a business partner fails to deliver what was promised or business growth requires extending operations, the workflow is repaired or augmented by searching for other providers.

The flow of control and information is not necessarily unidirectional—from negotiation to contract and from contract to workflow—but organically flows in both directions. For example, the workflow component might inform on its current operational capacity and production schedules in order to carry out informed negotiations and make utility-maximizing decisions. Also, if the inability to comply with the contract for any reason—production delays for example—is predicted, business partners can renegotiate to reach a new agreement, thus remediating the potential contract breach.

Similar business models have been proposed where contract drafting, formation, and fulfillment are sequentially occurring phases [11][33][39][47][34][63], however they do not consider the bi-directional relationship among e-commerce phases.

Based on this model, we survey the academic literature in the e-commerce domain—which includes e-contracting, negotiation, workflow, and business processes—for reference architectures, domain models, systems, and languages which exhibit some form of support for the integration of negotiation, contracts, and business workflow e-commerce components. The information or search phase is currently outside the scope of our inquiry.

The contribution of this survey is twofold. First, we propose a multidimensional framework which allows categorizing and evaluating the literature within the e-commerce context in a structured way.

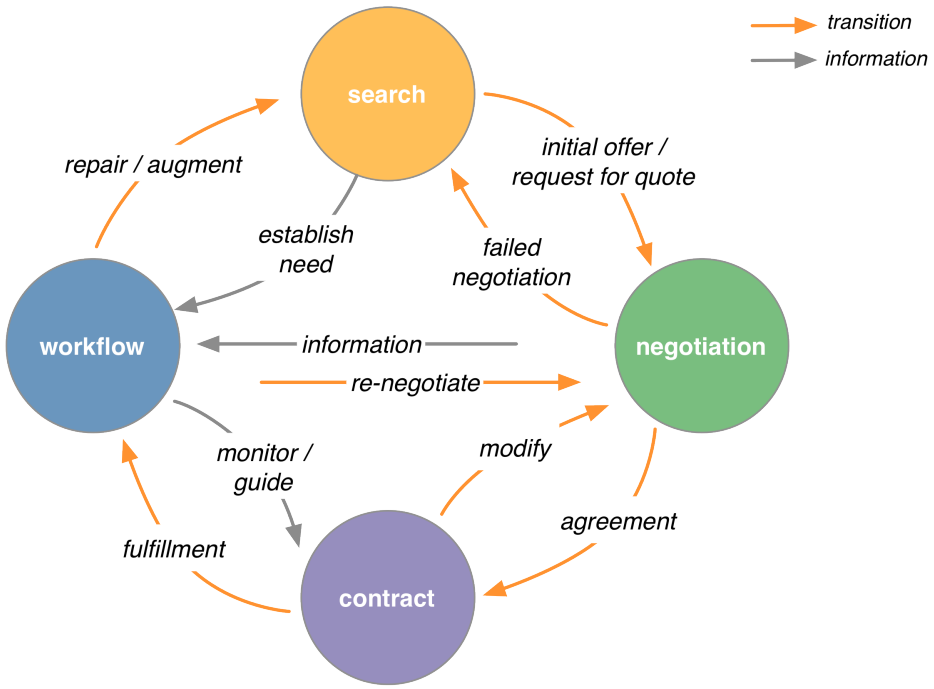

Fig. 1. E-commerce contracting lifecycle

On one dimension our framework assesses the synergic relationship between negotiation, contracts, and business workflow, analyzing the extent to which these components are integrated and bi-directionally interoperate within e-commerce applications. On the other dimension, it assesses the level of support of desirable architectural properties such as automation, dynamic adaptation, decentralization, exception handling, and security for each of the aforementioned e-commerce phases.

Second, we use this evaluation framework to survey the literature and analyze a representative set of e-commerce applications, providing insights on the exhibited integration of negotiation, contracts, and business workflow e-commerce components, as well as the degree of support of the functional and non-functional properties of interest.

The value of this study is identifying the shortcomings in existing research with respect to the described integrative approach to e-commerce, learning from existing innovative ideas, and identifying avenues for improvement of e-commerce systems. Our goal is to promote research on the e-commerce applications of the future, where inter-organizational business activities and global commerce are driven by dynamic coalitions among autonomous, decentralized, highly automated, and adaptive systems which support secure business relationships.

Conducting research on the e-commerce domain is of outmost importance for several reasons. First because the Internet has made possible the proliferation of global markets. Second, critical and private information is manipulated in e-commerce processes. Third because it is a multidisciplinary field. Lastly, because e-commerce is now ubiquitous.

Following, we define the terminology used throughout this paper (section 3). In section 4 we provide a brief characterization of the ongoing research within the negotiation, contracts, and workflow domains, including an overview of the relation between e-contracts and contract law. We present our research questions and describe the scope and the limitations of our inquiry (section 2). In section 5 we supply an account of related surveys. We present our multidimensional evaluation framework (section 6), a concise description of the evaluated studies (section 7), and the evaluation results (section 8). Finally, we discus our findings and insights (section 9), identify areas for future work (section 10), and provide concluding remarks (section 11).

2 Research goal, scope, and limitations

2.1 Objective

Our research is concerned with three important phases of e-commerce: the negotiation period where individuals and organizations reach an agreement, the contract which is established as a result, and the processes and activities that take place to fulfill the contract.

Despite the significant number of studies existent on each of these topics, very few concentrate on the synergy, intersection, and smooth transition between these phases. For instance, B2B and e-commerce frameworks have a focus on the exchange of meaningful business information within the workflow phase, but leave aside negotiation and binding contracts. In addition, these studies do not explicitly address and evaluate relevant architectural properties to provide insight on how e-commerce systems are being built and what are the consequences of their design decisions.

Our research question is how are negotiation, contracts, and workflow integrated in e-commerce technologies, and how are desirable system properties such as decentralization, dynamic adaptation, automation, security, and other relevant properties supported in each of these phases.

We are not only concerned with sequential integration of these e-commerce components—from negotiation to contracts and from contracts to workflow—but also on their bi-directional dependencies.

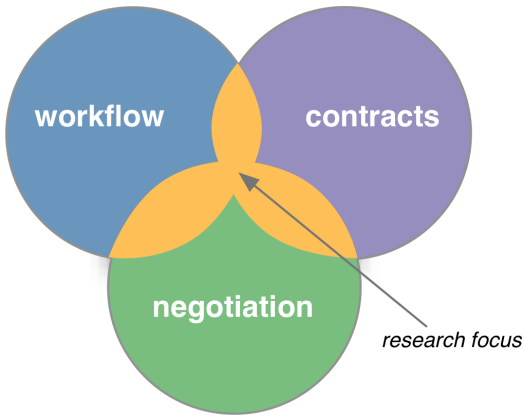

Fig. 2. Research focus

In order to answer our research questions, we propose a multidimensional evaluation framework (section 6) which provides a structured method to assess the relevant domain literature. We conduct a qualitative research methodology and base our findings on the properties of interest—outlined in section 8—on peer-reviewed journal publications and conference proceedings in the field of Software Engineering, Computer Science, and Business.

2.2 Survey scope

Our survey involves negotiation, contracts, and workflow within the scope of the e-commerce domain. While there are many aspects of e-commerce that can be analyzed with diverse goals in mind, we have chosen to focus on aspects that we consider appropriate for assessing the integration and dynamism across the e-commerce stages and components. Specifically, our evaluation framework emphasizes the integration and interoperation of negotiation, contracts, and business workflow components, as well as e-commerce systems' capabilities to dynamically adapt, automate processes, act autonomously, and provide security.

In this survey, we do not strictly categorize these studies as technologies for business-to-business markets. While B2B scenarios are more common within this literature and provide a more interesting perspective—given the complex interactions that take place among business partners—there are some B2C scenarios where negotiation does take place—such as auctions—therefore it is appropriate to include them in this discussion.

This survey assesses reference architectures, domain models, systems, and languages which address and support at least two of these e-commerce phases—negotiation, contracts, and/or workflow. Despite that there are many interesting studies in each of these areas which provide insight on particular concerns, we narrow the research scope to consider only those studies which exhibit some intersection of these areas and technologies, and provide insight on how they work together from an architectural perspective to provide greater support for the e-commerce life cycle.

The choice of evaluated approaches is by no means exhaustive, but it is a representative set of technologies that reflect the academic state-of-the-art in this field, the existing integration and interoperation approaches, and the overall adaptation, automation, and security provisions in this context. Evaluated approaches involve agent systems, workflow management systems, business processing, contracting systems, negotiation and auction systems, and e-commerce systems. We focus on overlapping concerns and integration efforts within these communities.

Although the information phase of e-commerce—where consumers look for potential suppliers or sellers—is an important stage within the e-commerce life cycle, we have not included it within our framework for two reasons. First, there are numerous ways in which a consumer can find commercial partners such as accessing supplier directories, leveraging recommendation systems, catalog browsing, or ad hoc manual processes. Given this diversity, it is difficult to assess the properties of interest without specific approaches to consider. The information phase is subject to specialized studies on the topic—such as in Schafer et. al [97]. Second, not including this phase is a research scoping decision given that just few of the surveyed studies address the information phase or include it within their proposed architectures. However, it is possible to extend our framework to include this phase.

Our research on contracts excludes work involving design-by-contracts [8], where a contract refers to an application interface and the pre- and post-conditions to use a component's functionalities. We are particularly interested on the representation of contracts as commercial agreements between parties.

Lastly, despite some of the analyzed architectures leverage service oriented principles, this survey does not specifically involve and is not limited to Service Oriented Architectures (SOA) [106]. Today many e-commerce systems follow SOA's principles of exposing system's functionality through interface services. However, our work goes beyond particular architectural styles, patterns, and programming paradigms to embrace the wide range of research exhibiting the properties we are concerned about.

2.3 Survey limitations

There are a few limitations to a more detailed and comprehensive study on this perspective to e-commerce systems. While there are commercial applications—such as eBay, IBM MQ Workflow, Staffware, COSA BPM, and Oracle Contracts—that would allow us to assess the "in practice" relationship between the e-commerce phases, we found no architecture description available or source code that we could analyze. The properties that we are interested in are hard to assess without such access to the architecture details and are not always discernible with product use. This is the reason why we focused on research papers that described the architectural properties of interest.

Furthermore, it was difficult to assess some of the analyzed architectural properties—such as the type of connectors used—without the availability of implementation artifacts or a detailed description of the architecture in research papers, at least with respect to the information we are concerned with. There are important aspects that are relevant for our study about which authors remained silent, either because the particular system does not support the features of interest or because they were not within the scope of their research. The properties that are important for the domain and scope of this research are not necessarily what is of interest to the authors of the selected studies in many cases. This left some questions unanswered with respect to specific studies.

3 Definitions

Software architecture is the set of principal design decisions made about a software system [106], including architectural elements—components and connectors—their configurations, and the properties—functional and non-functional—that these configurations bring forth.

An *architecture style* is a named collection of design decisions that apply to a given development context, constraining the set of design decisions that can be made and eliciting beneficial properties is a software system [106].

Evaluation framework. An evaluation framework—in this context—is a structured way of categorizing and analyzing architectures, systems, methodologies, or other technologies against defined criteria which constitute the evaluation perspective.

Integration according to Gulledge is "the interfacing of systems together so they can pass information across a complex technology landscape" [43]. We ground on this definition our own contextual meaning of integration and interoperability as the capability of a systems' architectural elements—components and connectors—or of entire systems to be part of larger systems and to communicate with each other, exchanging services and information through program interfaces. In this paper, we will refer to integration both at a technical level—where negotiation, contract, and workflow software components are linked and communicate—and at a higher conceptual level where concerns of all phases exist and are addressed within the same technology, including the smooth transition from one phase to the next.

Dynamic adaptation is the run-time modification of a software system to satisfy new requirements or changing circumstances [106].

Automation refers to processes and tasks—previously performed by humans—which are carried out by autonomous and self-driven software programs.

Negotiation is the ongoing conversation between self-governing parties to reach a mutually beneficial agreement. In a commercial transaction, the main criteria to negotiate are usually the nature of the exchanged goods, quantity or use period, price, and time for delivery.

Contract. A contract is a legally-binding agreement between parties engaging in an commercial exchange or transaction. For this discussion, the meaning of contract refers to a business agreement and not to an application interface as in the design-by-contract programming model [8].

Workflow refers to sequential or concurrent activities coordinatedly performed by individuals, groups, or systems in order to achieve a goal. Workflow is broadly used to refer to either a business process, a process specification, process automation software, or coordination and collaboration systems [33].

Business unit is an entity that has a specific purpose within a business or organization. Business unit can refer to a team with an assigned function or responsibility, a department, or a piece of software performing a specific business activity.

Component. In the narrative of our study we use the word component with two closely related but different meanings. Component refers to a constituent part of some broader concept or entity. For instance, when we refer to negotiation, contract, or workflow as components of e-commerce we intend to say that these are fundamental phases or elements of e-commerce. The second meaning of component refers to an architectural element which encapsulates data or functionality provided through explicit interfaces, and from which software systems are composed [106]. To differentiate these two interpretations of component we address to the latter one as software component.

4 Background

4.1 Negotiation

Negotiation is the decision-making process which involves the ongoing conversation between two or more parties with the intention of reaching an agreement that benefits all participants. Often, the main criteria subject to negotiation in a commercial transaction are the nature of the exchanged goods, quantity or time for use, the price, and the delivery time and terms.

Negotiation is an important component of e-commerce systems, mostly for business-to-business markets where complex interactions among supply chain partners take place. Although in business-to-consumer markets prices are usually fixed, auctions are a particular type of negotiation applicable also to both B2C and C2C markets where a product or service is sold to the highest bidder in a defined period of time. A popular example is eBay, a web-based platform for product offering and bidding.

Research on the negotiation domain involves negotiation mechanisms, auctions and other types of negotiation, automation, agent-mediated e-commerce, among others.

4.1.1 Negotiation mechanisms

A negotiation mechanism is the combination of a negotiation strategy — the sequence of actions during negotiation — and a protocol — the rules of interaction [68].

Negotiation strategies — mostly based on based on decision and game theory principles [85] — can be for example cooperative or competitive [44], or based on the exchange of persuasion arguments among negotiating parties [88]. Negotiation protocols have been developed to support bilateral and multilateral negotiations, as well as auctions [102][47].

4.1.2 Auctions

Although negotiation is more common in B2B scenarios — as opposed to business-to-consumer ones where prices are usually fixed — auctions [65] are a particular kind of negotiation which support both business-to-consumer and consumer-to-consumer negotiations, where a product or service is sold to the highest bidder on a predefined deadline. eBay, for example, provides a web-based platform for product offering and bidding.

4.1.3 Automated negotiation

A goal long desired — and the focus of many studies in the last 15 years — has been achieving automated negotiations [93] to build more agile, dynamic, and autonomous e-commerce systems. Digital agents negotiate on behalf of humans and organizations by collecting data, evaluating offerings, searching for deals, and making utility-maximizing decisions, hence optimizing the negotiation process. However, in today's e-commerce, humans are still responsible for evaluating offers and making decisions [69]. Negotiation is a complex problem — both for humans and systems — where social, economic, legal, and management factors interplay.

4.1.4 Agent-mediated e-commerce

Agent-mediated e-commerce research [69] is — to great extent — a dominant topic within automated negotiations. An agent is a purposeful software component which is capable of performing autonomous and reactive actions within a given deployment environment. Agents are considered appropriate for negotiation since they capture the autonomous and self-interested nature of the individuals and organizations they represent. Aside from negotiation, agents have been leveraged for product and merchant brokering, and buyer coalition formation [47]. In addition, mobile agents have been used to support the design of location-aware and mobile e-commerce applications [61].

Agent-mediated e-commerce architectures and prototypes include auction servers and applications [115][95], marketplace-based bilateral and multi-lateral negotiations [16][21], multi-attribute negotiations [69][46], and merchant comparison mechanisms [69].

In addition, this body of work includes technologies and languages that implement the aforementioned systems, such as the Electronic Data Interchange (EDI) language for inter-organizational communication, domain or system specific XML-based languages (eg. ebXML and XrML), web ontologies (e.g. RDF, DAML, OIL), and agent communication languages (e.g. FIPA-ACL, KQML) [47].

4.2 Contracts

A business contract involves a set of statements describing the exchanged goods or services, as well as the permissions and obligations held by participating parties — individuals or organizations — contingent on a sequence of events or contract fulfillment steps. These business rules are often scattered between physical and digital documents, databases, and systems.

With an increasing number of commercial activities carried out online, supporting the collaborative establishment of contracts that can be monitored and executed by e-commerce applications is necessary. Contracts are the foundation to support the integration and automation of application components implementing commercial transactions [40].

Formalizing complex, natural language contracts abounding in legal terminology into expressive machine-readable languages that semantically capture domain-specific agreements continues to be a challenging problem [62]. Several questions arise with respect to "e-contracts" such as:

- what is the essential information that needs to be captured and what can be left behind without affecting contract execution?
- what is the appropriate degree of detail of captured information?
- how can e-contract formalism express agreements that are semantically meaningful for a wide range of domain-specific organizations with organization-specific terminology?
- are there any important clauses that cannot be formally modeled?
- how to handle unknown data and events unforeseen at contract time?

Despite these challenges and open questions, the expected benefits of e-contracts are eliminating natural language ambiguity, identifying conflicting clauses, enabling contract execution automation, increasing productivity, providing inter-agency accountability and awareness, enabling hypothetical reasoning, guiding business workflow, among others.

Research on e-commerce contracts has focused on their explicit formalization. Different flavors of logic-based formalisms have been leveraged, such as deontic logic [72], defeasible logic [37], courteous logic [41], μ-calculus [86], event calculus [30], or a combination of them [38][87]. Usually these logic-based formalism are then translated into some machine readable notation, often XML-based.

Less formal approaches have also been proposed such as the explicit semantic relationship between natural language statements [25], Petri Net-based contracts [23], and finite state machines representations which explicitly model temporality and the acceptable sequence of events [24][79].

Additionally, research has been done in exception handling or "contrary-to-duty" specifications contingent on contract violation or non-compliance [38][24]. Furthermore, contract monitoring [66][79], enforcement [79][78], and execution [37][40], as well as prototyping have also been in this domain's research agenda.

4.2.1 Service level agreements

A Service Level Agreement (SLA) [55] is a specific type of contract in which the service provider commits to supply an acceptable quality-of-service (Qos) to the service consumer. This type of contracts are appropriate for quantifiable services or service properties such as bandwidth, recovery time, performance, latency, jitter, and so on. For example, the help desk can take no more than 24 hours to hours to respond to a product maintenance inquiry, or a customer's Internet connection bandwidth should be no less than 30 Mbps.

4.2.2 E-contracts and Law

The rapid growth and pervasiveness of e-commerce ceased to be exclusively a matter of technological innovation, becoming a topic of interest within legal discourses. Electronic contracts span concerns from multiple disciplines such as business, economics, logistics, and computer science. Law—and specifically contract law—is specially concerned and tightly related to e-contracts. Contract law "determines the enforceability of the promises of the parties and is the body of law applicable to the formation, interpretation, and performance of the contract, as well as for the remedies in the event of the failure of a party to perform the promise(s) made" [91]. E-contracts are unprecedented to a legal doctrine that is originally based on interactions among individuals.

For e-commerce to be a feasible and scalable business branch within organizations—especially in B2B markets—it is necessary for e-contracts to be legally binding to increase trust among parties and be instruments to legal dispute resolution.

There are two fundamental concerns with respect to e-contracts and Law. The first concern is how do local and international legal frameworks embrace e-contracts and e-commerce activities, and how can e-contracts be law abiding. The second is how to specify e-contracts so they are valid and complete from a legal perspective.

Even before e-commerce as we know it, lawyers analyzed the legal implications of technology such as EDI within businesses [27]. Diverging opinions arose on whether computer-generated agreements should be enforced as legally binding contracts [4] and if so, how electronic contracts would fit within the current legal system and be dealt with within courts. Legal experts have argued that contract law does not need to change since structurally paper contracts and e-contract are equivalent, and thus contract laws apply equally [49]. Allen et. al [4] with a more progressive approach state that it is not a matter of whether law should change, but how should it change to embrace technology and innovation.

Many legal researchers have adopted the latter position where law needs to adapt to current demands and address novel issues [59]. Such adaptation can be based on existing law theory and practice [113], on similar situations previously addressed by the legal system [59], by making required modifications and additions to the legal framework embracing unprecedented situations [59] [5], or by creating new legal frameworks specific for e-commerce [49].

E-contracts have been discussed from many perspectives, raising challenging questions and concerns such as whether e-contracts can be traded as other valuables, the liability and the legal consequences of automated e-commerce activities, and whether individuals and organizations will eventually cease having direct knowledge and contact with trading partners [4].

Today, electronic contracts are as legally binding as paper contracts through state and federal laws such as the Uniform Electronic Transactions Act (UETA), the Electronic Signatures in Global and National Commerce Act (ESIGN), the European Union's E-Commerce Directive, and similar laws in other countries. Under these laws, the action of electronic agents and automated e-commerce systems are legally bound to a liable person or organization. The law excludes certain types of contracts such as family documents—wills, divorce, adoptions, and so on—real estate, and safety or health related—such as insurance [1]. These exclusions suggest lack of trust in electronic documentation for critical situations, which along with mistrust in communication media have been all along the main barriers for e-commerce adoption [34]. Also, we speculate that e-commerce is reluctantly adopted in countries where the proper legal framework for e-commerce does not exist or is not well established.

Digital signatures—replacing pen and paper ones—certify that the contract was signed by an authorized person or an organization's representative, and that the contents of the document have not been forged along the way. Digital signatures are based on Public Key Infrastructure (PKI), a cryptographic technology currently considered the most reliable method for signing and providing access to authorized parties to e-contract's content [1] and guaranteeing non-repudiation against unforeseen denial or contract violation threats.

E-contracts must fulfill a set of requirements to be legally enforceable, for example clearly identify parties involved, the description of the contract subject, the period of contract validity, and the agreement of parties specified through digital signatures [4][34]. Besides these general requirements, the contents of a contract will depend on the terminology and practice of the specific application domain.

Despite guiding e-commerce laws, many challenges remain such as the differences in the legislation support across countries [5] (as well as the lack of support in others), transactions that span multiple jurisdictions, the recognition of a contract in other countries, the degree of confidence in automated e-commerce activities, and the extent to which humans are responsible for their effect.

A more technical concern with respect to e-contracts and Law is whether specified e-contracts are valid and complete from a legal perspective, namely whether contract specification languages allow complying with these legal requirements. As briefly described in section 4.2, many logic-based contract formalisms have been proposed. Deontic logic has been widely considered to describe contracts given its support for concepts such as obligation, permission, and prohibition [112]. Extensions to basic deontic logic have been proposed to include, for example, provisions for non-compliance (contrary-to-duty)[13] and to resolve conflicting contractual clauses [37].

E-contracts—and e-commerce in large—embody multidisciplinary concerns that span business, law, economics, marketing, information technologies, computer science, and so on. Although e-commerce research in the Computer Science and Software Engineering fields does not entirely address these multidisciplinary concerns, there is some work done which includes, for example, economic and legal aspects of e-contracts [77][34][5].

The legal support for e-contracts—through national and international legislations—is critical for the growth and evolution of e-commerce. The benefits of such support from a business perspective are significant, including efficient information management, integration with suppliers, just-in-time procurement, lower transaction costs, and opportunity to participate in broader markets [22].

4.3 Workflow

Workflow refers to the set of interdependent and coordinated tasks carried out by individuals, organizations, or computers to achieve a goal. To fulfill business, commercial, and organizational goals, agencies share information, carry out co-dependent activities, and strive for the integration of business units. Workflow technology attempts capturing these tasks where humans and software cooperate to achieve specific goals. The expected benefits are efficiently supporting business operations, increased productivity, increasing activity awareness, and reduced operational costs through the smooth integration and automation of intra- and inter-agency business processes. The field's challenges include:

- mapping high-level business goals to specific tasks performed by a network of people and systems;
- the smooth integration of heterogeneous systems and data models;
- the automation of intra- and inter-organizational business processes;
- a decentralized and collaborative fine-grained workflow specification;
- flexible and dynamic adaptation to changing requirements.

4.3.1 Workflow specification languages

Workflow specification languages formally describe the process by which tasks are performed, addressing task dependencies, data format and exchange, error handling, and recovery [64]. These languages differ in the expressiveness to model workflow patterns [110], their underlying theoretical foundations, their scope, or on whether they are document-, activity-, or participant-centered. Examples of these languages and standards include declarative rule-based languages (e.g. TSL, WFSL [64]), Petri Net-based (e.g. YAWL [109]), web-services-based (e.g. BPEL [60]), based on process calculus and concurrent processing (e.g. BPML [60]), graphic languages (e.g. BPMN [60]), UML-based [26], to name a few.

4.3.2 Workflow management systems

While workflow specifications focus on process description, workflow implementation addresses process execution and monitoring. A workflow management system (WfMS) allows the specification, execution, and monitoring of workflows, and facilitates human coordination, collaboration, and co-decision [33].

WfMS — alike workflow specification languages — are document- (e.g. Xflow [71]), activity- (e.g. InConcert [73]), or team-centric (e.g. Team Automata [28]), or a combination of these (e.g. ObjectFlow [50]). Early WfMS were mostly document-centric.

Other concerns in the WfMS domain are the distinction between transactional and non-transactional tasks (e.g. Meteor [64]), workflow migration and exception handling (e.g. Adome [20]), and workflow linkage (e.g. CrossFlow [39]).

In recent years, research has shifted from general-purpose WfMS to domain-specific ones, exploring their application in a variety of domains such as software development, emergency planning and contingency, medical image processing, staff training, among others.

4.3.3 Business process management systems

Business process management systems (BPMS) involve technologies to design, execute, and analyze business processes [110]. BPMS — successors of WfMS — focus on higher-level business concerns as opposed to low-level operational processes. However, many nominal BPMS have basically the same functionality as traditional WfMS. Some researchers argue that the distinction between BPMS and WfMS is the inclusion of a process diagnosis and re-design phase in BPMS [110][60]. This phase embraces dynamic markets where organizations need to adapt to changing requirements. Examples of BPMS are FlowMark [67] and ARIS [98].

4.3.4 E-commerce systems and frameworks

With the increasing popularity of e-commerce — and enabling open markets — there has been a growing venture on securely supporting B2B transactions. E-commerce systems are distributed, decentralized, and dynamically created inter-organizational workflows [107] within Internet trading communities. Examples of e-commerce systems are eCo System [35], Commerce XML, and BizTalk [100].

4.3.5 Communication and data exchange standards

Non-standardized business terminology and the heterogeneity of business applications pose a barrier to interdepartmental and inter-organizational information exchange. To remedy this problem communication and interchange languages and business integration standards have been the underpinning of B2B collaboration. For instance, Electronic Data Interchange (EDI) is considered as one of the precursors of e-commerce by allowing data exchange—such as invoices and purchase orders—through strictly formatted messages. Current e-commerce is increasingly moving away from EDI-based communication and towards XML and XML-based languages (e.g. ebXML)—although EDI is still used in some industries—given its human interpretability. Interchange languages such as the XML/EDI attempt to represent EDI messages in XML [35]. Aside from these domain independent languages, domain specific standards have been proposed to enable communication in particular domains. For example the Open Trading Protocol and SWIFT for the banking and payment industry, RosettaNet for the electronics manufacturing industry, Open Buying on the Internet (OBI) for large-scale procurement, Information and Content Exchange (ICE) for content syndication, Open Financial Exchange for financial statement exchange [35], ebXML for B2B communication, cXML for procurements, among others.

5 Related surveys

Hvitved [51] extensively surveyed formal languages and models to represent contracts, which enable automated validation, execution, analysis of contracts. This study compares formalisms by a set of contract requirements, such as their capability to describe contract participants, conditional clauses, absolute and temporal constraints, exceptions, and so on.

Angelov et. al [6] describe a set of functional and non-functional requirements for e-contracting systems based on software architecture and e-contracts literature.

Electronic negotiation approaches have been surveyed from different perspectives and through different classification criteria. Bichler et. al [9] point out the lack of guidelines for choosing appropriate negotiation mechanisms for particular contexts, and propose a multidisciplinary engineering approach for creating negotiation models. In addition, this study provides an overview of some of theoretical foundations behind surveyed information systems. Kersten et. al [56] provide a historical overview of supported and automated negotiations, making a explicit differentiation between negotiation support systems — passive support of social systems — and e-negotiation systems — active participation of socio-technical systems. Buttner [12] focuses on automated negotiations and provides a classification taxonomy by which automated negotiations are categorized by their structure, their process, their theoretic foundations, and their restrictions.

Studies have surveyed the workflow domain — including business process management — providing an overview of current workflow methodologies and systems, as well as future research directions [33][7]. Technologies to support inter-organizational workflow automation as well as integration of heterogeneous, autonomous, and distributed e-commerce systems are discussed in these studies.

Van der Aalst et. al [110] provide a historical perspective, a description of emerging technologies, and the state of the art of business process management. The authors underline the need for clear scientific foundations and more formal approaches to methods and languages for the evolution of business process automation. A more specialized description on business process management languages and standards can be found in [60], classified by their support of process execution, interchange, graphical representation, and diagnosis.

E-commerce frameworks support inter-organizational communication and online transactions. Shim et. al [100] provide an analysis of popular B2B e-commerce frameworks with specific concern on interoperability and security. Their findings show that these frameworks still lack required support for quotation handling and the semantic conversion of ontologies.

Various researchers have surveyed the role of stationary and mobile agents within B2C as well as B2B e-commerce [45][47][61]. Agents were found to play roles on need identification, product and merchant brokering, negotiation, product evaluation, buyer coalition formation, location-aware and mobile shopping, and auction bidding. In addition, areas for research and application of agents within B2B e-commerce have been identified — sourcing and procurement, workflow, supply chain, and supplier relationship management — to enable integration and automation [10].

Although these studies are comprehensive and have provided great insights on the state-of-the-art and the challenges within their own areas, our survey approach encompasses the integration of negotiation, contracts, and workflow. A recurrent argument found in these literature is the need of multidisciplinary approaches towards e-commerce and business processing. We take a fundamental step in this direction by analyzing the level of support current technology provides for these synergic and co-dependent components of e-commerce — negotiation, contracts, and business workflow. In addition to our integrative approach towards surveying these areas, we strongly focus on decentralization, distribution, dynamic adaptation, automation, and security aspects which are fundamental for the scalability, optimization, trust, and evolution of e-commerce systems.

6 Multidimensional evaluation framework

Our evaluation framework grew organically from our interest in the e-commerce domain and specifically on how to describe computer-processing commercial contracts exchanged and agreed upon among autonomous and distributed parties. From our initial inquiries, we found the significant literature overlap and the tight relationship of this domain with both negotiation and workflow.

Our evaluation framework is a multidimensional one, which on one dimension assesses the existence and the way e-commerce components—negotiation, contracts, and workflow—relate, integrate, and interoperate, and on the other dimension it evaluates if and how important system properties are supported on each of these phases. Figure 3 presents the dimensions of our framework.

The evaluation criteria were chosen based on careful reflection and reasoning about what are the fundamental functional—what the system does—and non-functional—abstract system qualities—properties [106] that e-commerce systems need to support. Some of these criteria assess the type of party interaction supported and how system components exchange information. Others instead assess system qualities such as decentralization, automation, dynamic adaptation, and security.

Our evaluation criteria are grounded on previous research which has emphasized their beneficial properties such as scalability, reliability, performance, trust, and efficiency of software systems [83][54][70][58][90], which are appropriate in the context of the e-commerce domain.

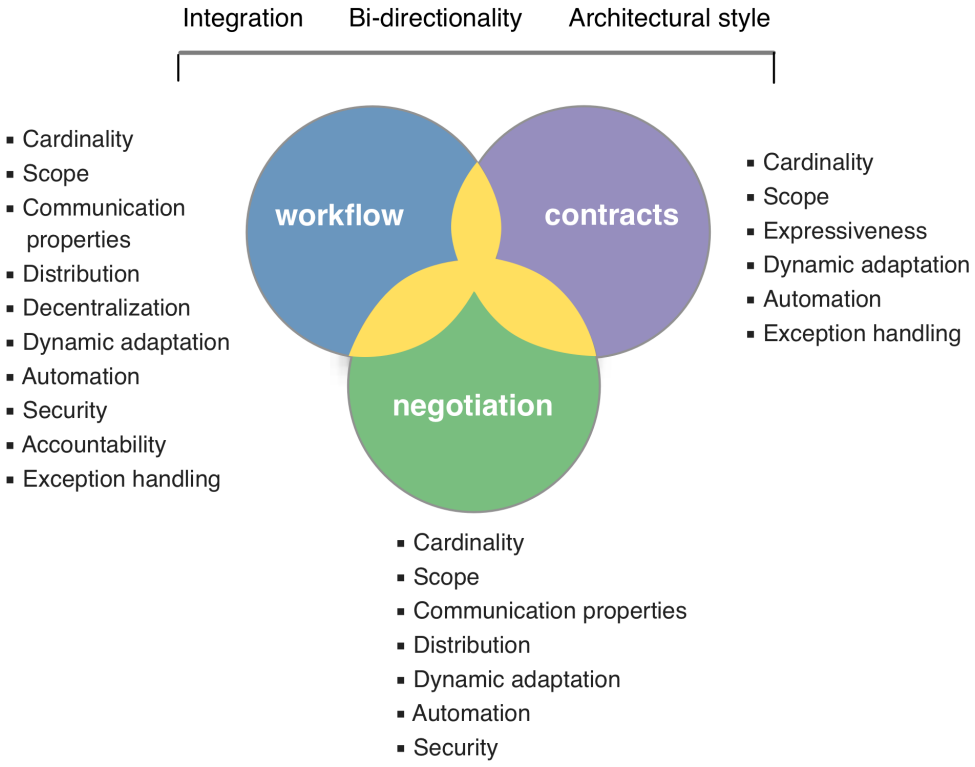

Fig. 3. Multidimensional evaluation framework

6.1 Negotiation

6.1.1 Negotiation cardinality

Negotiation cardinality assesses two properties with respect to negotiation: *interaction cardinality* and *cardinality of negotiation attributes*. Interaction cardinality refers to whether the evaluated technology supports one-to-one, one-to-many—such as auctions—or many-to-many negotiating parties. Cardinality of the negotiation attributes assesses the expressiveness of the negotiation, namely whether the system supports negotiating on a single issue or multiple issues.

These criteria are based on Lomuscio et. al's negotiation classification scheme [68], where they are called "interactions" and "negotiation domain" correspondingly.

6.1.2 Scope

Negotiation scope describes the aspects in a given context that a system's architecture addresses [6], specifically whether a system's negotiation mechanisms:

- are specialized to support a particular domain;
- provide extensions to support a domain's data model and terminology;
- or, provide general purpose and standardized support for negotiation.

Negotiation scope determines how applicable and appropriate a negotiation component is for a particular design problem.

6.1.3 Communication properties

We assess how negotiating parties communicate through two specific properties: *software connectors* and *data format*.

Connectors enable interaction, control, and data transfer among software components [75]. This assessment is important given that connectors are largely responsible of system scalability, performance, reliability, security, and reuse, as well as provide insights on the architectural style—or styles—the systems follows. With this purpose, we leverage the connector taxonomy provided by Mehta et. al [75], where connectors are categorized by the service they provide — communication, coordination, conversion, and facilitation — and by their type — procedure call, event, data access, linkage, stream, arbitrator, adaptor, and distributor.

The data format communication property describes the type and structure of the information exchanged among software components.

6.1.4 Distribution

Negotiation distribution assesses whether negotiating components or processes are running in different execution locations (i.e computers). This assessment is relevant to system scalability, negotiation latency, and fault tolerance.

6.1.5 Dynamic adaptation

Dynamic adaptation is the capability of evolving a software system by reconfiguring its architecture at run-time. In the negotiation discourse, we are interested in assessing whether the evaluated system supports dynamic adaption with respect to:

- the negotiation strategy of participants according to the attitudes of their negotiation counterparts or to newly acquired information;
- the negotiation location — for example in a scenario where participants negotiate in a centralized marketplace — or the location migration of any of the negotiation participants.

Very few studies are concerned with the adaptation of negotiation strategies or negotiation location. Oprea [82] provides a negotiation model by which agents learn from the interaction with other negotiating agents — namely they model the other agent's negotiation strategy — to improve agents' negotiation competence.

6.1.6 Automation

Automation involves the delegation of tasks to software components [106] and systems, requiring less or no manual intervention, therefore optimizing processes and increasing business throughput. Software components are able to make autonomous decisions based on their provided business logic.

In our context, automation assesses the degree to which a system's negotiation components reach an agreement on behalf of the represented parties through a sequence of self-driven decisions. On one end of the spectrum — where automation is not supported — negotiation is carried out by humans through manual processes. On the other end, all negotiation activities are automated through interacting software agents that carry out the whole negotiation process on behalf of its owners [47].

6.1.7 Security

Security assesses whether the analyzed system provides security and privacy mechanisms to the negotiation process or to the negotiation participants.

There are two perspectives to security during negotiation. The first perspective is concerned with the various levels of trust, non-repudiation, and information privacy in the face of unknown negotiation partners. Security in this context protects negotiation participants and their private and strategic information against fraud and untrusted parties. An example is a security model where negotiation parties do not need to reveal private constraints on the terms of the agreement—business operations or strategies—in order to reach an agreement [31].

The second perspective refers to the security of the communication medium against identity theft and malicious eavesdropping parties. Security mechanisms to hinder these threats come in different flavors, such as encryption of messages, certification, custom authorization mechanisms, and so on.

6.2 Contracts

6.2.1 Interaction cardinality

Interaction cardinality refers to the number of parties involved in a contract. In other words whether the evaluated system supports the creation of bi-party or multi-party contract specifications.

6.2.2 Scope

Scope assesses the capability of the system—or its underlying contract specification language—to specify specific types of contracts. In other words, whether the contracting language is particular to the specification of contracts in a given domain or whether the contracting language or infrastructure is general purpose, and therefore allows the specification of contracts in a variety of domains.

6.2.3 Expressiveness

Expressiveness refers to the richness of the contract data model. In other words, it is the degree to which the contract specification supports formalizing multiple issues and concerns with respect to the obligations and permissions of participating parties. This criterion provides as well insight on the tradeoff between expressiveness, difficulty, and accuracy of formal notations for specifying contracts.

6.2.4 Dynamic adaptation

Dynamic adaptation assesses whether the contract—currently guiding the activities of participating parties—can be modified and re-enacted at run-time to accommodate a new agreement or to be augmented with additional clauses.

6.2.5 Automation

With respect to business contracts there are four contracting stages that can be automated: contract establishment, analysis, enactment, and monitoring. Contract establishment automation refers to parties' capabilities to collaboratively formalize their agreement. The automation of contract analysis involves conflict detection within the contract. Enactment and monitoring automation refer to self-driven contract execution by participating parties and the run-time checking of conformance to contract clauses, correspondingly. The framework assesses automation from two different—but related—perspectives:

- how amenable to automation is the underlying contracting formalism;
- whether the evaluated system supports — and how — the aforementioned forms of contract automation, with the exception of contract enactment which is delegated to the workflow automation criteria included in this framework (see section 6.3.7).

6.2.6 Exception handling

Exception handling assesses whether the contract model or the contracting language supports the specification of contrary-to-duty statements [38][24], namely alternative courses of action contingent on contract violation by either of the participating parties.

6.3 Workflow

6.3.1 Interaction cardinality

With respect to workflow we assess whether the evaluated system supports the design and execution of inter-organizational workflows, or if it is limited to support activities within a single organization.

6.3.2 Scope

Scope in the context of workflow informs whether the evaluated system supports the flow of activities specific to a particular domain or in contrast whether the system is domain independent, thus supporting any type of business endeavors.

6.3.3 Communication properties

We assess how workflow components communicate through two specific properties: *software connectors* and *data format*. Similar to the communication properties within negotiation (section 6.1.3), we assess which are the kind of software connectors used to allow communication among components performing different workflow activities and what is the format of the exchanged information.

6.3.4 Distribution

System distribution with respect to workflow includes:

- *activity distribution* which refers to whether the system supports the coordination and execution of large-scale workflow activities performed by collaborating distributed — and potentially heterogenous — software components which belong to either a single or multiple parties;
- *workflow management distribution* assesses whether the process is being executed by a single, centralized workflow engine, or whether the workflow is executed among distinct communicating workflow engines, thus promoting scalability and decentralization [81].

6.3.5 Decentralization

Decentralization allows components under different authorities to make their own decisions, providing and requesting services for their individual purposes and interests. A distributed system is not necessarily decentralized, since distributed components can be centrally controlled by a single authority.

The framework assesses a system's support for decentralized interaction of business processes both at intra- and inter-organization levels. The former refers to business unit or sub-process autonomy within the same organization. The latter to whether the system enables inter-organizational workflows.

Furthermore, decentralization in this context is extended to decentralized workflow management, namely allowing parties to control their corresponding portion of the process within a workflow description. For example, Fakas et. al [29] propose a peer-to-peer architecture for decentralized workflow management, stepping away from intermediaries and centralized process control.

6.3.6 Dynamic adaptation

Workflow adaptation can take place both statically — by modifying the workflow specification and restarting its execution — and dynamically — allowing run-time workflow management where work units change their course of interaction dynamically according to new workflow specifications.

Business process management systems have largely focused on providing the capability to dynamically adapt a process enactment. This criteria assesses the degree to which the system allows the dynamic adaptation of the workflow, where on the one side of the spectrum a system can be rigid and static, and at the other side accommodating and very flexible to change.

A workflow adapts according to different adaptation rules. For example, Sadiq et. al provide a taxonomy to describe workflow change, where current workflow instances are *flushed, aborted, migrate, adapted, and built* [92]. For the evaluation of the selected systems we are interested in assessing whether these kinds or any type of workflow adaptation to changing requirements is supported.

6.3.7 Automation

Workflow automation enables the self-driven execution of business processes by a software system according to a set of rules, optimizing the process, and reducing operational and administrative time and cost. Automation is a persistently pursued goal within workflow research [33][60].

We assess the extent to which a system supports workflow automation, ranging from no-automation support — fully manual and human-driven — to partially automated — requires some human input — to fully automated process execution.

6.3.8 Security

Sharing and interfacing with external systems and services — those which belong to other organizations — brings up many security issues and concerns within the e-commerce and e-business domain. The pursued goal is achieving the right balance between security and trust, as well as between privacy and information and process disclosure.

Similar to (6.1.7) there are two orthogonal aspects of security. The first is concerned with business security and protection against unreliable, unknown, or malicious business partners and fraudulent actions. The second is a more technical concern regarding the security of the communication medium.

There are a variety of security mechanisms available to support workflow, business processes, and e-commerce systems. Some systems may rely on all-purpose network security standards such as HTTPS and SSL, and others may choose a domain specific mechanism such as the approach to workflow views in Schulz et. al [99] which selectively hide details of private workflows, while providing interfaces for the allowed communication between trading partners.

We are interested in assessing the security provisions, both from a business trading perspective as well as from a technical one within the workflow or contract enactment phase.

6.3.9 Accountability

Workflow management systems are included in what Suchman [103] has termed as "technologies of accountability", where actions performed by individuals and organizations are visible—with custom authority restrictions—to others. Accountability is related to role and responsibility, where an activity is assigned to an individual or organization—which has an specific role in the workflow—and each party is accountable and responsible for the activities assigned.

Accountability is important for successful business relationships, mostly during contract enactment, where parties track the state of the business process for both trust and coordination reasons. Suchman makes the analogy between a bookkeeper's ledger and these technologies which—in this particular context—account for what activities in the contract have been completed and which are those outstanding obligations. This evaluation framework assesses if and how the studied reference architectures and systems support the accountability of actions and of contract state among business partners.

6.3.10 Exception handling

Exception handling is of utmost importance in the workflow domain to recover and resume interrupted business processes. Exceptions might be caused due to errors in the communication channels — for example network failure — or by the delay, failure, or non-compliance by any given business unit. Our framework assesses the exception handling and recovery mechanism provided–if any — by the evaluated system. For example, Agentwork [80] uses a rule-based approach to specify exceptions and their corresponding workflow adaptations.

6.4 Architectural style

An architectural style is a named collection of architectural design decisions that are applicable for a particular development context [106]. Identifying the particular style or the combination of styles of an e-commerce system allows assessing whether the set of properties the style brought forth provides the desired system qualities and benefits. In doing so we gain deeper understanding of the underlying rationale of certain decisions. If an architectural style is not explicitly mentioned, it is possible to infer the type of architectural style based on the used software connectors.

6.5 Integration

Integration is a common, but very broadly used term to denote stand-alone software components or systems that are brought together to compose systems of systems. However, integration embodies specific meanings depending on its in-practice application context and domain.

Gulledge [43] presents a taxonomy to classify integration. Under this taxonomy our focus regards point-to-point and enterprise application integration — for internal processes — as well as B2B e-commerce integration — involving inter-organizational data exchange. We are particularly interested in assessing the existence and concrete nature of the integration of three fundamental components of e-commerce applications: negotiation, contracts, and business workflow.

6.5.1 Integration of negotiation and contracts

Integration of negotiation and contracts refers to whether the end result of negotiation between two or multiple parties is explicitly a contract description or formalism.

6.5.2 Integration of negotiation and workflow

Negotiation and workflow are integrated when the result of the negotiation has a direct impact over the business workflow's configuration.

6.5.3 Integration of contracts and workflow

The integration between a contract and workflow refers to the conformance of the workflow configuration with the contract specification, where the workflow or set of workflows are described in such a way that fulfill their corresponding obligations within the contract. This framework assesses whether the contract formalism directly guides the workflow, or the contract is an executable instance of a workflow, or the workflow is generated from a contract description.

Furthermore, a change in the contract might require — run-time — adaptation of the workflow which fulfills it. Therefore we assess if the workflow changes based on run-time contract modifications.

6.6 Bi-directionality

If integration between negotiation, contract, and workflow components exists, then we assess whether the bi-directional flow of data and control among them is supported. Despite that the usual sequence of events is to progress from negotiation to contract formation, and then to the execution of business processes, bi-directionality involves going back to prior stages, or using or sending information to processes or components which belong to previous phases.

6.6.1 Bi-directionality between negotiation and contracts

Bi-directionality between negotiation and a contract is the ability of — not only going from negotiation to contract establishment as its product — but also being able to go back to a negotiation stage to modify or augment an existing contract.

6.6.2 Bi-directionality between negotiation and workflow

There are a two of ways in which bi-directional relationships between negotiation and workflow can exist. The first is when business partners—from an ongoing workflow phase—go back to negotiation, contingent on, for example, workflow delays that hinder a party to fulfill its timely obligations, thus requiring new agreements.

The distinction between bi-directionality of negotiation and contracts, and of negotiation and workflow is that in the former the existing contract artifact is modified, while in the latter that is not necessarily the case, potentially discarding the old contract instance and creating a new one.

The second form of bi-directional relationship has an informative purpose, where the workflow management informs the negotiation component about the business's operational capabilities, so better decisions can be made during negotiation and a profitable — and realistic — agreement can be settled.

6.6.3 Bi-directionality between contracts and workflow

A bi-directional scenario between contract and workflow might involve monitoring the current workflow based on the contract specifications in order to perform the required adjustments.

7 Overview of surveyed architectures

With the goal of surveying and evaluating the domain spanning negotiation, electronic contracts, and workflow, we selected a set of representative systems from the contracting, agent, workflow, negotiation, and e-commerce communities. These studies include reference architectures, domain models, systems, and languages, which we will be referred to as technologies. Although the artifacts of these contributions are different, we consider it fair to compare them given that the properties an evaluation criteria of interest are applicable to all of them.

Our selection criteria is based on whether the analyzed system architectures or frameworks address at least two of the areas of interest. Despite the availability of interesting research work on these areas, we excluded from our evaluation studies that are concerned on either contracting, negotiation, or workflow in isolation. These are for example negotiation systems—such as Persuader [105], AuctionBot [115], Kasbah [16], Tête-à-Tête [69], Aspire[57], and Dr-Negotiate [101]—contracting systems—for example SweetDeal [42] and the 4W Framework [5]—or workflow management systems—such as InConcert [96], ActionWorkflow [74], Meteor [64], ObjectFlow [50], Mentor [114], Team Automata [28], eCo [35], Mobile [52] and IBM's MQ.

For each technology a typical use example or algorithm is supplied. Examples and use steps have been fully or partially extracted from the research papers, and have been in some cases modified to provide shorter descriptions.

7.1 Reference architectures

A reference architecture is the set of principal design decisions that are applicable to multiple related systems within a domain, with explicitly defined variation points [106].

7.1.1 EC-Brokering and Notarial Service

EC-Brokering and Notarial Service (ECBNS) [84] (figure 4)—within the MEMO project—is a distributed workflow architecture with transactional semantics to support interoperable e-commerce transactions and e-contracts. This object-oriented architecture includes a market-place with browsable business partner catalogues, negotiation and contracting support, and notary services. This work describes an architecture design and a potential implementation, but no existing prototype.

Use scenario

1. A customer browses electronic directories and catalogs though an ECNBS broker's search engine looking for potential business partners.
2. The search engine accesses business profiles and products databases using standard domain terminology that match the search criteria.
3. When a provider has been selected, ECBNS's *negotiating and contracting manager* supports the collaborative specification of standard transaction terms.
4. The result is a final contract which is stored in a contract base at the broker's server.
5. The contract base is input to an EDI-based workflow manager which will coordinate the execution of the contract.
6. The workflow manager maps e-commerce transactions to cross-organizational business processes.
7. Through this mapping, the automation of processes begins.
8. Parties activities are recorded within a log for reliability and non-repudiation among business partners and to resume activities in case of system failure.

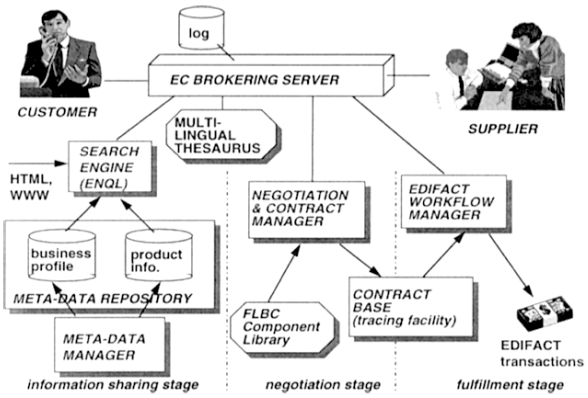

Fig. 4. ECBNS architecture [84]

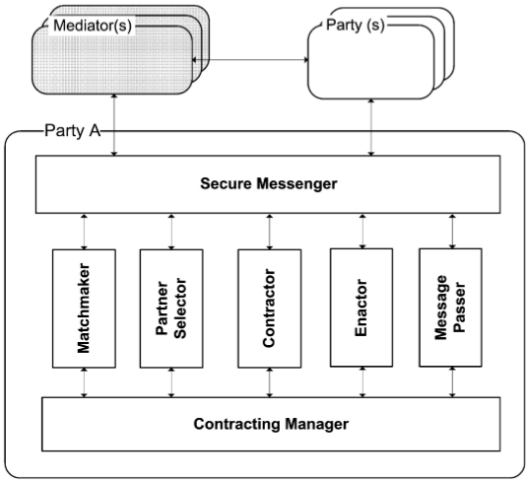

Fig. 5. E-contracting reference architecture [6]

7.1.2 E-contracting Reference Architecture

The E-contracting Reference Architecture (ERA) [6] (figure 5) provides a set of functional and non-functional requirements, as well as the architectural basis for developing modular, highly automated, component-based e-contracting systems. This architecture puts forward the essential set of components to support searching for business partners, negotiating, establishing, and executing e-contracts. These high-level components are a *matchmaker, partner selector, contractor, enactor, contracting manager, and secure messenger.*

Contracting algorithm

1. The contracting manager receives a request to establish a contract.
2. The contracting manager sends the matchmaker a "matchmaking request", to which the matchmaker publishes corresponding advertisements at external parties.
3. The matchmaker receives in return external advertisements matching the published ones.
4. Cross-referencing the published and received advertisements, the matchmaker creates a list which is returned to the contracting manager.
5. The contracting manager sends the partner selector a request, to which this sends "information requests" to selected parties.
6. The contracting manager sends the returned list of results to the contractor which creates a set of "contract offers".
7. If an agreement is reached with any of the parties, the contractor creates a contract.
8. The contracting manager sends a message to the enactor to start the contract fulfillment.
9. The enactor—similar to a WfMS—manages the value exchanges, monitors the contract, attempts to resolve disputes, and evaluates activities.

7.2 Domain models

Domain models are those which do not provide an specific reference architecture or system, but prescribe domain elements, requirements, and algorithms particular to an application domain.

7.2.1 Business Contract Architectural Framework

Milosevic et.al [77] conducted one of the earliest studies which considers the legal, business, and technical aspects of e-commerce. It attributes the slow adoption of e-commerce to inadequate representations

of business semantics and the lack of legal support. The Business Contract Architectural Framework (BCAF) (figure 6) provides a high level contract architecture which puts forth requirements for e-commerce systems such as contract domain specification, contract templates, negotiation, validation, monitoring, and enforcement.

Model description

1. A contract domain is established which defines the boundary in which a contract is valid, can be disputed, and enforced.
2. A contract template specifies parties roles, contract period, exchanged goods, and obligations.
3. The contract template is input to negotiation, where parties resolve conflicting interest and reach a mutual agreement.
4. The agreement contract is then validated against rules of the contract domain.
5. Contract monitoring—maybe through a third party—involves observing parties' behavior.
6. Contract enforcement ensures behavior conforms to the contract or dictates corrective actions.

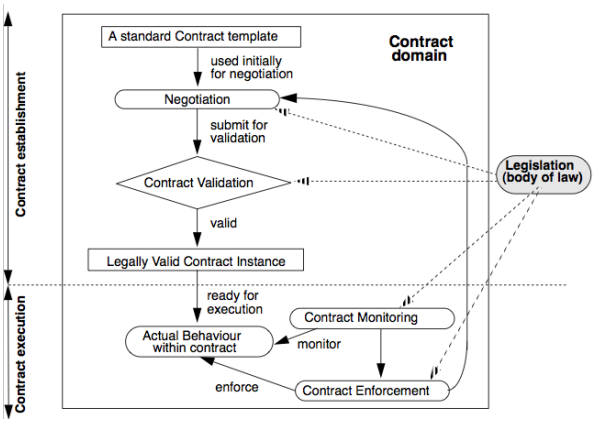

Fig. 6. Business Contract Architectural Framework [77]

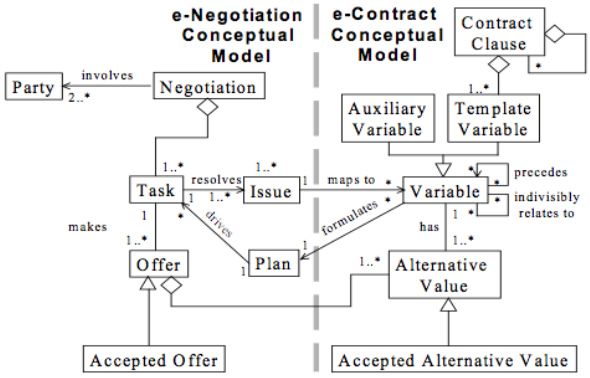

Fig. 7. e-Negotiation - e-Contract concept model [17]

7.2.2 e-Negotiation based on e-Contract meta-model

Cheung et. al [17] (figure 7) propose a meta-model for e-contracts based on contract templates and their negotiable variables. This model builds on the relationship among negotiation variables—as opposed to single-issue negotiation—including trade-offs, order of precedence (to negotiate variables), and variable grouping. From the relationship between variables a negotiation plan–namely an algorithm—can be derived. The authors suggest building on top of the E-ADOME workflow technology [18] to carry out negotiation tasks according to the established plan.

Contract negotiation algorithm

1. The negotiation initiator identifies issues that require some service or solution, and the criteria adopted for the negotiation process (e.g. minimum sell price is $100).
2. Concurrently a contract template is selected, which is a set of clauses which may recursively contain contract variables which will be defined during negotiation.
3. Then the set of identified issues are mapped to template variables in the contract template.

4. Variables —such as rent and period in a lease contract—are ordered by precedence, thus defining which variables are to be defined before others. A variable might *indivisibly relate* to another, therefore they are to be negotiated together.
5. The variable relationships are verified against the negotiation requirements.
6. A negotiation plan—specifically an acyclic graph—is derived from the variable precedence and indivisibility relationships, so that variables in a vertex G can only be negotiated after their parents.
7. Offers and counteroffers are made according to the negotiation plan. Variables are bound to values.
8. The process ends with a finalized e-contract if all the negotiation tasks are successful.

7.3 Systems

7.3.1 Agent Enhanced Workflow

Judge et.al [53] (figure 8) propose an agent-enhanced approach to mitigate shortcomings of workflow systems by adding a software agent layer atop of a commercial workflow management system. Agents allow the WfMS to react to changes in the work environment, increasing automation. A prototype of this architecture has been applied to—but is not limited to—a correspondence handling centre.

Use scenario - correspondence handling domain example

1. A company receives correspondence from its customers regarding different issues.
2. Correspondence is handled by the company's correspondence handling center, composed by a central administration (CA) and potentially outsourced work processing centers (WPC).
3. The CA receives, classifies, searches for contractors, and distributes correspondence among WPCs.
4. The CA invites WPC to bid for portions of the advertised categories of work—namely handling a specific category of correspondence.
5. The CA assigns work to bid winners, thus establishing a binding contract with the selected WPC.
6. The CA monitors the overall solution by maintaining local copies of contracts with WPC.
7. The CA selectively excludes successful bidders for the next round of work auctions.
8. A WPC can refuse to respond to the bid invitation if it cannot process additional work.
9. A WCP can also overbid to advertise its processing capabilities.
10. The process continues until all the work has been distributed.
11. If a WPC cannot process the assigned work, it is returned to the CA and auctioned again to all WPCs excluding the incapable WPC.

The correspondence handling system is implemented by a WfMS with a CA agent and multiple WPC agents which manage the central administration and the work processing centers correspondingly.

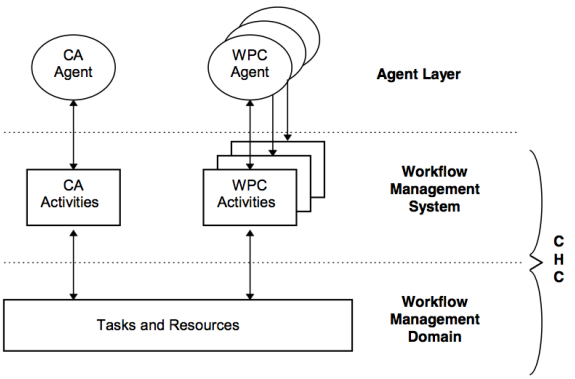

Fig. 8. High Level System Architecture [53]

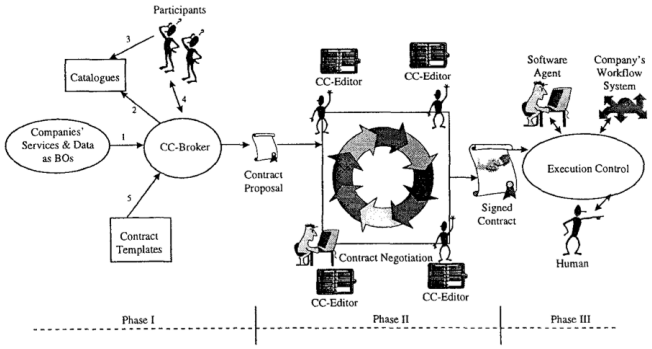

Fig. 9. COSMOS Reference Architecture [40]

7.3.2 COSMOS

COSMOS [40] (figure 9) is an Internet-based electronic contract service, which provides online catalogues and broker services for finding business partners, and enables creating, negotiating, and executing contracts. The infrastructure relies on business objects—application domain entities such as people, products, departments, etc.—and adapters which provide logical access points to an organization's business components, minimizing the impact on internal data and processes.

Use scenario

1. The COSMOS service is hosted by a COSMOS provider which may be a bank, a public authority, a notary, or other trusted facilitator
2. Brokers at the COSMOS service act on behalf of customers to find potential contractors for a specific service type (interface type).
3. The broker accesses the COSMOS service to look for a service provider or product vendor in the offers catalog according to the customer's QoS requirements and constraints.
4. Providers register services in the catalog by specifying its service type along with the corresponding *adapters*—a logical access point to a company's business objects.
5. If the requested service matches any of the offerings, references are returned to the customer.
6. Once a provider has been selected, COSMOS supports the collaborative editing—namely negotiation through offers and counteroffers—of a contract template guided by a negotiation protocol.
7. Business objects—goods or services—are included explicitly in the contract object.
8. Notarial services are provided through a signing support, which employs digital signatures and verifies all participants signed the same document version.
9. A distributed Petri Net-based workflow representation is generated from the contract.
10. The contract is executed by the COSMOS workflow engine, which coordinates parties' activities.
11. The contracted services are accessed through the *business objects' adapters* included in the contract.

7.3.3 MAGNET

MAGNET [21] (figures 10 and 11) is a market architecture which supports multi-agent negotiation. This virtual market is an intermediary between provider and supplier, monitoring the contract execution, and increasing trust. MAGNET allows engaging in many specialized market sessions at the same time. For example, a company might engage in negotiations within several specialized domain markets to find suppliers for completing different task in its workflow.

Use scenario - Acme Software example

1. Acme Software has the goal of produce and ship software packages.
2. Acme Software's intelligent agent acts as a broker within an *exchange*, which is a set of domain-specific markets. A *market* is a commerce forum where buyer and selling agents interact.
3. The Acme agent has the goal of selecting one or more product/service markets and establish one or more contract to fulfill the company's business goals.
4. The agent selects both a publishing service market and a packaging and shipping market.
5. The agent uses ontologies provided by the markets to formulate a business plan to fulfill its service or product needs.
6. The plan is entered into the Magnet *client* interface as a Gantt chart along with a property sheet to editing task parameters.
7. The Acme agent initiates *sessions* in the selected markets and submits calls-for-bid in each session to start negotiations.
8. The session invites registered providers to participate by joining the session, and interested providers bid for the service/product contract.
9. The customer agent evaluates the bids and awards the contract to one or more contractors.
10. The business plan is executed and upon activity or activities completion the session is terminated.
11. The agent monitors and repairs its business plan when a provider fails to deliver by creating a new session and negotiating a new contract.

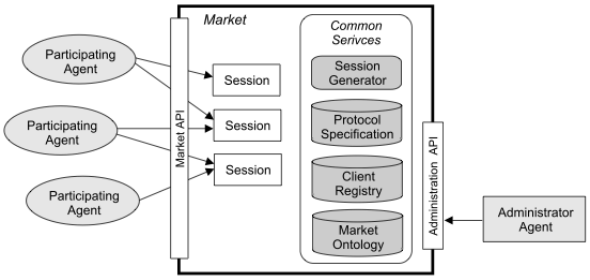

Fig. 10. The Market [21]

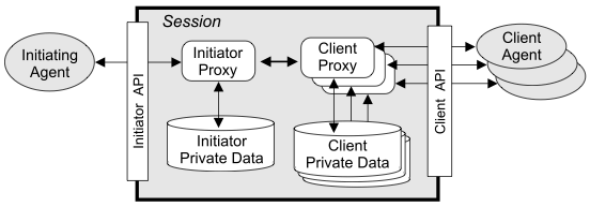

Fig. 11. The Session [21]

7.3.4 CrossFlow

The CrossFlow [39] (figures 12 and 13) project supports cross-organizational workflows based on electronic contracts which describe the cooperation of virtual enterprises created through dynamic service outsourcing. CrossFlow approaches the dynamic linking of workflow management infrastructures to achieve service enactment, transaction management, fine-grained process control, and quality of service monitoring. CrossFlow bridges the e-commerce and workflow domains through a common service consumer/provider paradigm.

Use scenario

1. An organization that wishes to outsource some of its business activities queries a matchmaking facility or marketplace to find appropriate service providers though a *contract manager*.
2. Matchmaking is based on demands of the contracting party and service descriptions within contract templates which are advertised by service providers.
3. The consumer's contract manager selects a service provider's template matching its needs which is completed into a binding contract proposal.
4. The contract is sent to the provider for acceptance or rejection of the service terms and price.
5. At the contract approval, a dynamic service enactment infrastructure is configured symmetrically at both consumer's and provider's CrossFlow systems.
6. The infrastructure provides cooperation and coordination support, and safe proxies-gateways for cross-agency communication.
7. The service consumer can outsource the process through the configured proxies. Outsourcing is performed dynamically during the execution of the activities requiring the service.
8. A service consumer may request modifications to the parameters of the outsourced service.
9. The service consumer is notified of the completion of the service. Upon satisfaction of the service provided, the dynamically created infrastructure is dynamically disposed.

7.3.5 B2B Contracts with BizTalk

Herring et. al [48] leverage Microsofts BizTalk infrastructure to create and exchange XML-based contracts. The architecture includes a contract repository, a notary, a contract monitor, and a contract enforcer. Contracts are the basis for generating policy documents and a business plan which dictate the behavior of business objects executing the contract.

Use scenario - music trading example

1. Musac.com wishes to sell its music though eShop's web portal.
2. To do so, Musac and eShop use B2B.com's standard business documents—such as contracts and purchase orders—to facilitate trading activities. B2B's client software runs on eShop's server.

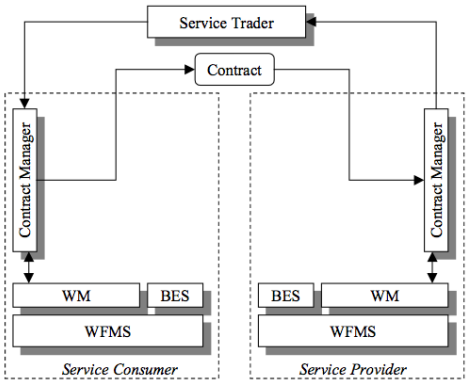

Fig. 12. CrossFlow - contract establishment [39]

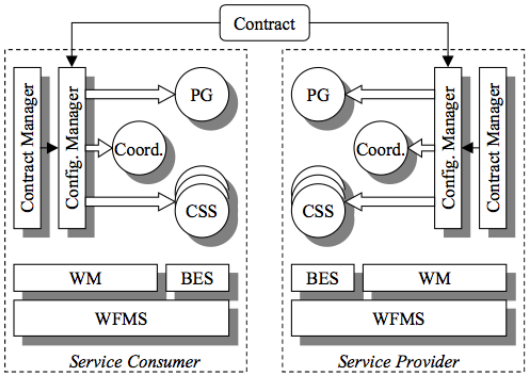

Fig. 13. CrossFlow - dynamic configuration [39]

3. A Musac employee fills a contract form at eShop's website, digitally signs it, and submits the form.
4. eShop processes the contract, approves, and signs it automatically.
5. The contract is stored in the *notary* component at B2B's server.
6. B2B's *contract management* software automatically creates a set of *business policy documents*—which are refinement of high-level contract clauses—based on the contract instance, the corresponding template, and generic rules applicable to that template.
7. Policy documents are sent to both Musac and eShop to guide their behavior during contract enactment, i.e. obligations, permissions and prohibitions.
8. From policy documents, *business plan documents* are generated for each of the participating parties, which specify the activities that need to be carried out to fulfill the contract.
9. These actives are mapped to business objects that will implement the required behaviors.
10. The business transaction then begin, allowing, for example, Musac to transmit music files to eShop's server and for eShop to send invoices to Musac accounting department.
11. All transactions are monitored by both parties, as well as B2B.

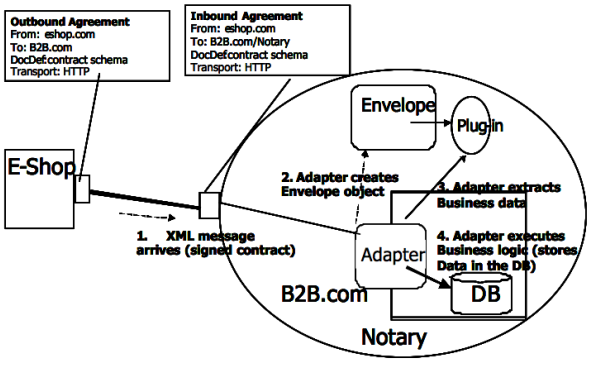

Fig. 14. BizTalk infrastructure - notary component [48]

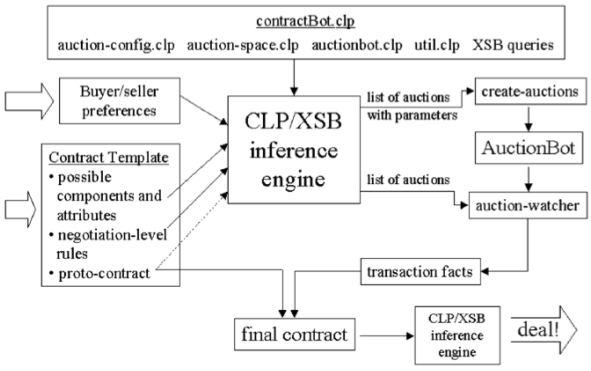

Fig. 15. ContractBot - executable contract creation [89]

7.3.6 ContractBot

ContractBot [89] exhibits a very distinct flavor of contract establishment by automatically creating a set of auctions given a set of domain-specific parameters and negotiation preferences. These bids are contract templates executed by an underlying auctioning infrastructure—AuctionBot. ContractBot monitors auction results to finally establish an executable contract between business partners. A rule-based approach to contracts is expected to provide substantial execution automation.

Use scenario - travel packages example

1. A travel agency's agent is delegated the responsibility of assembling travel packages, and buying and selling travel services through an auction server.
2. There are three types of goods: flights, hotels, and entertainment tickets, which require different auction configurations due to their distinctive parameters and business requirements.
3. The contract template of such travel package contains possible negotiation parameters, their relationship, and the set of rules by which each of the goods ought to be negotiated. The template might include also rules from potential domain specific buyers and sellers, for example hotel rooms are sold individually or in blocks.
4. Based on this contract template, ContractBot infers auction parameters and configures the set of auctions that will be instantiated in AuctionBot:
   - Flights are sold at randomly-fluctuating fixed-prices with date, inbound, and outbound destination parameters.
   - Hotels are sold in a variant of ascending English auction with day and quality parameters.
   - Show tickets are sold in stock-style double auction with day and event type parameters.
5. With the auctions results, ContractBot completes the travel package contract.
6. The contract is an executable that is input to the ContractBot's workflow engine.

7.3.7 HP Contract Framework

Focusing on business-to-business markets, HP Labs developed a contract framework [11] (figures 16 and 17) to support the whole contract lifecycle based on contract formation—which includes collaborative drafting and negotiation mechanisms—and contract fulfillment. In addition, a basic syntax for expressing electronic contracts is provided based on deontic operators.

Use scenario

1. Parties that wish to negotiate, establish, and automate their contractual relations, have installed and connected backend systems to the HP Contract Framework.
2. Parties have their Contract Frameworks connected through a messaging mechanism.
3. A contract template is leveraged to start the negotiation, and its exchanged among parties through the *Contract Negotiation Protocol* to make counteroffers, accept, or reject the contract proposal. The protocol allows parties changing contract variables and adding or removing clauses.
4. Once parties agree on clauses and variables, the template is signed establishing a finalized contract.
5. The enactment of the contract is carried out through a *Contract Fulfillment Protocol* which enables meaningful communications.
6. In the framework, a *reasoner* analyzes the contract and according to normative statements establishes the activities to fulfill.
7. Progress and fulfillment of the activities are determined by the analysis of incoming—from the local systems—and outgoing messages.
8. Based on new information, the reasoner computes the state of the contract and determines the following activities that need to be carried out.
9. These activities might be carried out by local systems—or an action executor—or a request for service might be sent to other business partners through their connected frameworks.
10. Parties can access a *document event store* to view documents created, sent, and received.

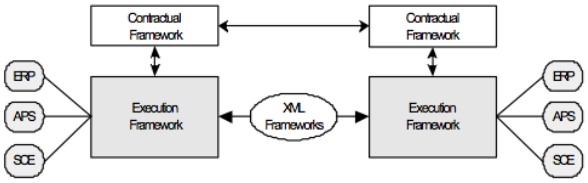

Fig. 16. HP Framework - distributed deployment [11]

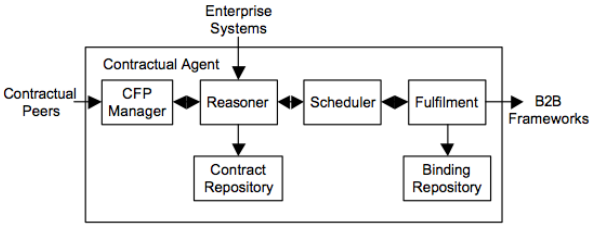

Fig. 17. HP Framework - conceptual architecture [11]

7.3.8 E-ADOME

E-ADOME [18][19] (figures 18 and 19) is a cross-organizational workflow environment based on the concept of *workflow views*, subsets of a workflow definition, which deliberately reveal or conceal information to balance trust and security between trading partners. E-contracts include formal descriptions of views and communication graphs between them (input and output messages). An agent interface is provided for specifying, executing, and monitoring e-services, as well as for exception handling. This work includes the inter-operational workflow model based on XML, Web services, and the ADOME workflow engine.

Use scenario - Internet service re-seller example

1. Dickson Computer Systems (DCS) provides leased-line Internet service packages.
2. A typical workflow starts with a user who wishes to contract an Internet service by filling out a web-based form on DCS's web page and registering a domain name as part of the service package.
3. The contracted package includes as well a server PC, its installation, an a customized website.
4. These services are automated through DCS's WfMS.
5. Many of the workflow activities are outsourced or goods are purchased to third parties. For example, the server is purchase to a PC vendor and the leased-line from another provider.
6. Given all these different interactions that DCS has with the customer, the PC vendor, and the leased-line provider, its WfMS interacts with its partners' WfMS.
7. The customer can access DCS's service package information and the order progress information and is notified on any service or delivery delay.
8. However, customers cannot access source and price of goods that DCS obtains from third parties.
9. The service/product vendors only reveal prices and technical specifications of their offerings. Updates on products or software are notified and DCS can monitor the work outsourced.

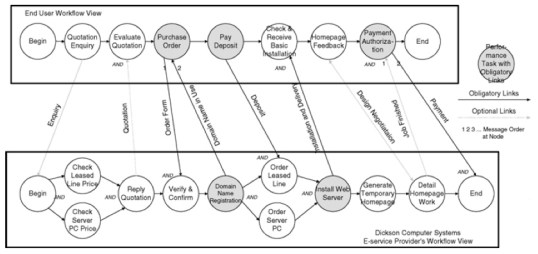

Fig. 18. An E-contract communication graph between two workflow views [18]

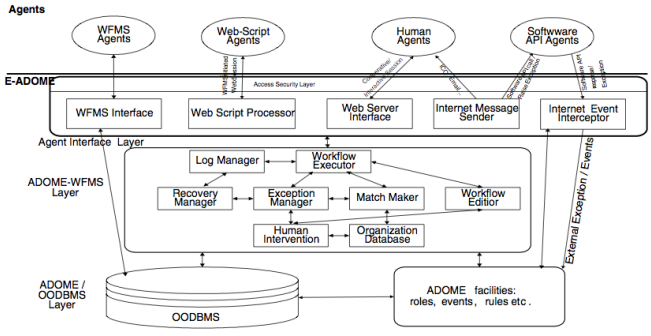

Fig. 19. E-ADOME architecture [18]

7.3.9 eMediator

eMediator [95] takes a modular approach to e-commerce by providing three main components: eAuctionHouse—a configurable auction server—eCommiter—a contract optimizer—and eExchangeHouse—an exchange planner. The e-commerce server makes strong use of game-theoretic incentive mechanisms—to maximize participants' payoffs—during the negotiation process and contract commitment.

Description - eAuctionHouse

- eAuctionHouse is an web-based auction and exchange prototype to be hosted by a trusted party.
- eAuctionHouse supports a variety of combinatorial auctions where users bid on combinations of products or services.
- It also support combinatorial exchanges where there are multiple buyers and sellers concurrently. This means that in a single bid a participant can both buy and sell items.
- eAuctionHouse allows bidding via price-quantity graphs for multiple indistinguishable goods.
- eAuctionHouse support the user choosing the most appropriate auction type via an expert system.
- Supports bidding through different types of mobile agents.

Description - eCommiter

- Assumption is that not always parties are benefitted by a full contract commitment given future events that could be more beneficial.
- Supports leveled commitment contracts, which allow a party to decommit from the contract unilaterally and pay decommitment penalties.
- eCommiter optimizes a contract by calculating the contract price and decommitment penalty for each party in a way that maximizes the sum of parties' expected payoffs.

Use scenario - eExchangeHouse

- eExchangeHouse plans peer-to-peer goods exchange in settings where a single item, multiple independent items, or multiple dependent units are being exchanged.
- Consider a software vendor selling a software which involves a main package and a plugin, therefore a total of 5 dependent cd-roms.
- The input to eExchangeHouse are two graphs, one for the buyer and one for the seller, on how the value increases as a function of how much has been delivered.
- The planner then finds a way to split the goods into pieces which minimizes the number of pieces and generates a safe delivery sequence if it exists.
- The exchange between buyer and seller proceeds according to the plan.

7.3.10 ER^{EC} Framework

This e-contract framework [63] supports modeling, storing, managing, enacting, and monitoring contracts. ER^{EC} maps XML contract descriptions to implementations based on WfMS and web services. Consequently, this contracting component can be part of larger systems and interoperate with the contracting parties' SOA-based systems, facilitating cross-organizational interaction. ER^{EC} also enables the analysis of what-if scenarios of e-contract clause violation.

Use scenario

1. The scenario starts with a contract which is already specified in an XML document.
2. Using the ER^{EC} framework system the user inputs *activity-party-clauses (APC)* constructs, which relate clauses in the contract, to responsible parties, and to concrete activities.

3. Then, using the ER^{EC} system the user also inputs activity commit diagrams, which specify the order of each activity and alternative activity paths. This and the previous step are done manually.
4. The user identifies phrases in the contract containing "if then else", "contract violates", or "but", and using the ER^{EC} system inputs a set of *event-condition-action (ECA)* rules through identifying the set of events and the corresponding actions that need to take place accordingly. Events include exceptions and actions their corresponding handling. ECA rules can also be generated based on APC and ACD specifications.
5. A set of workflows are generated by the ER^{EC} system based on APC and ACD specifications. The tool allows modifying the generated workflows.
6. Workflows are carried out through a WfMS and cross-organizational activities are triggered via web services.
7. During execution the ER^{EC} *contract enactment monitor* tracks committed activities and validates them against the specified activity commit diagrams for consistency.
8. ECA rules are evaluated during the execution of each activity so that alternative workflow paths can be taken and exception can be handled.

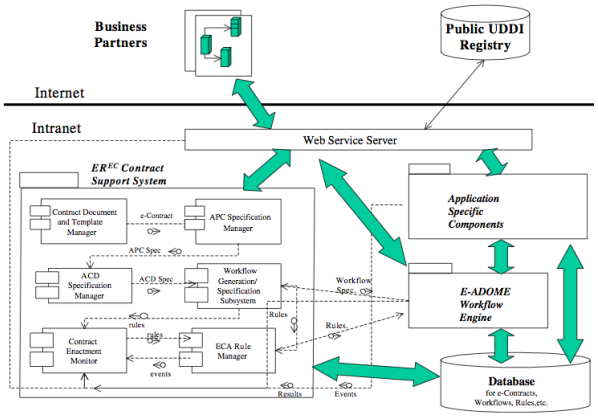

Fig. 20. Architecture for ER^{EC} framework [63]

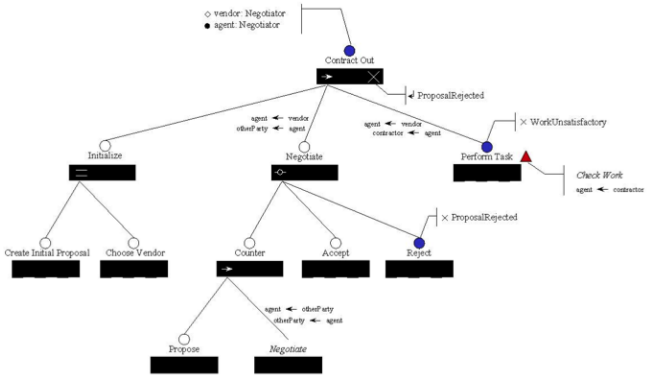

Fig. 21. A Simple Negotiation Process [15]

7.4 Languages

7.4.1 Little-JILContract Negotiation

Little-JIL [15] is a graphical language for agent coordination, where processes are decomposed hierarchically into a set of connected "steps" which are assigned to either a human or software agents. Its strict graphic semantics promote understandability, ease of use, and unambiguity for specifying the coordination of processes. This language has been proposed to model the coordinated interaction of negotiation participants and enable automated negotiations [14]. The advantage of this approach is that the negotiation process is explicitly described as opposed to being fixed within an implementation. Also, a library of contracting process in Little-JIL is presented. For the purposes of brevity we will refer to this approach as "Little-JIL" in the remainder of the paper, although this language is not specific to contract negotiation.

Use scenario - one-to-one negotiation example

1. Two parties that wish to negotiate collaboratively specify the negotiation process.
2. A hierarchy of *steps* are specified and the following sequence of actions can take place:
   (a) On the top level the sequential *contract out* step is specified, with three children steps: *initialize*, *negotiate*, and *perform task*.
   (b) The *initialize* step involves describing the task to be negotiated—thus creating an initial proposal—and selects the agent to perform the service.
   (c) The *negotiation* step begins by sending a proposal to the service provider, to which three alternative steps can happen: *counter*[offer], *accept*, or *reject* the proposal.
   (d) If the *counter* step is carried out it carries out a *propose* step to create a proposal or counter-proposal and then cyclically returns to the *negotiate* step.
   (e) If the *reject* step is carried out by one of the parties, the whole process starts by returning to the *contract out* step.
   (f) If instead the offer is accepted though the *accept* step, the process progresses to the *perform task* step.
3. For each step the agent that carries out the activity is specified.
4. The outcome process description guides negotiation agents in coordination and communication activities, and can be executed by the Juliette interpreter.
5. Little-JIL also allows specifying in a similar way processes for different types of auctions such as sealed bid and open-cry auctions.

8 Architectures evaluation

Following, we provide an assessment of the evaluation criteria with respect to the evaluated systems. Find in section 8.7 a complete version of the evaluation results for each study.

In the tables presented along the following discussion certain are labeled with "n/a" or "n/s" labels meaning "not available" and "not specified" correspondingly. Not available means that the e-commerce technology—this being a reference architecture, a domain model, a system, or a language—does not support or address the corresponding criteria. Not specified means that while the study suggest some level of support for the corresponding property, it is not specific enough to make an specific categorization and assessment.

8.1 Negotiation

In this section, we assess the type of negotiation supported (cardinality), whether it supports a specific domain (scope), the software connectors and data format used (communication properties), whether it is a distributed system, and whether is supports dynamic adaptation, automation, and security mechanisms. The main findings within this area are:

- The most frequent negotiation type is the auction.
- Many-to-many negotiations—aside from double auctions—are not supported.
- Auctions generally negotiate a single attribute such as price.
- One-to-one and many-to-many negotiations are flexible in the number of negotiated attributes.
- Most negotiations are domain independent and negotiate unbound variables in contract templates.
- Ontologies and domain specific rules and attributes are leveraged to include domain knowledge.
- The most common connectors are distributor connectors (network protocols), remote procedure call, and event connectors. Others are data access and complex custom connectors.
- Most common data exchanged are method arguments and XML-based documents.
- Parties negotiate distributedly through intelligent agents, distributed negotiation infrastructures, or through intermediaries.
- Dynamic adaptation of negotiation strategy is not supported and rarely that of negotiation location.
- Some systems exhibit partial or full automation for negotiation and contract formation.
- Security—surprisingly—is not a central topic within contract negotiation.
- Third party mediation, digital signatures, and encryption are the most recurrent security measures.

8.1.1 Negotiation cardinality

Interaction cardinality is a fundamental property of negotiation which defines the purpose or goal of the negotiation, hence applications implementing the various types of negotiations are fundamentally different. Analyzed systems support one-to-one, one-to-many, and/or many-to-many negotiations.

- *One-to-many* negotiations—namely auctions—are the most frequent type of negotiation. Auctions have been leveraged as a method to distribute tasks among a set of work units [53] or as a buying and selling mechanism [21].

- *Many-to-many* is the least frequent type of negotiation found in this literature. This excludes intermediaries and brokers, since their role is assisting in negotiation, but not actively participating. eMediator provides a model and prototype for combinatorial double auctions where there are multiple buyers and sellers concurrently. The meta-model proposed by Cheung et. al [17] also describes this type of negotiation, but does not provide concrete implementation or application details. This kind of interaction is rare, which illustrates the challenging nature of multiple parties engaging in negotiation of the same contract. ERA suggests a related approach where there are many one-to-one concurrent negotiation threads to find the best trading opportunities.

Table 1. Negotiation - interaction cardinality

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/s	one-to-one	one-to-one	one-to-one / many-to-many	one-to-many	one-to-one	one-to-many	n/a	n/a	one-to-one / one-to-many
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				one-to-many	n/s	n/a	one-to-many many-to-many	n/a	

With regards to the cardinality of the negotiation attributes, auctions are in general negotiated over a single attribute such as price. Instead, studies whose architectures have not been identified as auctions, but as negotiations between two or more parties, exhibit more flexibility in terms of the number of issues or attributes that can be negotiated. Attributes negotiated are—for example—price, delivery date, QoS attributes, or other negotiable contract parameters. So there seems to be—in general—a consistent relation between one-to-many to single negotiated attribute, and a relation between one-to-one and many-to-many to multiple attributes.

Table 2. Negotiation - cardinality of negotiation attributes

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/s	multiple	n/s	multiple	single	multiple	n/s	n/a	n/a	n/s
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				single	multiple	n/a	single	n/a	

8.1.2 Scope

The evaluated architectures fit within a broad spectrum of domain support, ranging from domain-independent technologies to exhibited negotiation support for a particular domain.

- Domain independent systems negotiate unbound parameters within contract templates [11], QoS offer attributes [40], combinations of products or services [95], or any type of offerings [77][6].
- A middle-ground approach is to provide general purpose negotiation mechanisms, but with the flexibility of extending these to encompass domain specific concepts. ContractBot, for example, allows configuring auctions with additional domain specific rules and attributes. MAGNET provides markets—negotiation intermediaries—with ontologies to support negotiation in specialized domains. The contracting model by Cheung et. al allows modifying or augmenting standard contract templates to include specific business interactions. ECBNS uses standard terminology to search for products in a particular domain within a meta-data repository, however it is not clear whether these domain terms play a role within negotiation. A particular case is Little-JIL, where domain specific agents implement the behavior specified in a domain independent negotiation model.
- On the other end of the spectrum, negotiation supports a particular kind of contract. For example, the Agent Enhanced Workflow (AEW) architecture [53] is specific to auctioning work items among a set of task-performing agents—with the assumption that many agents can perform the same task. This type of system is applicable to domains such as correspondence handling, call centers, and supply chain management.

Table 3. Negotiation - scope

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
domain independent	domain independent	domain independent	domain specific modifications to standard templates	work items bidding	QoS attributes	market provided domain ontology	n/a	n/a	domain independent negotiation process and domain specific agents
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				auctions based on domain specific rules and attributes	domain independent	n/a	domain independent	n/a	

8.1.3 Communication properties

Within the evaluated architectures we found distributor, procedure calls and remote procedure calls, event, data access, and event connectors, as well as complex custom connectors.

- The most common connectors within the analyzed architectures are *distributor connectors*, specifically network protocols such as http and TCP/IP. These connectors mediate the interaction between distributed parties, which in this context are negotiation agents and auction servers [53][89]. Network protocols are also leveraged to support more complex, application level connectors such as multicast mechanisms (distributor) to notify bidding parties about the auction state [53].
- *Procedure calls or remote procedure calls (RPC)* are widely used connectors, mostly given that negotiation agents—or mobile agents as in eMediator—are often designed in an object oriented style [53][95]. In MAGNET remote method invocations—the Java version of RPC—are used to interact with the negotiation server. The data subject of communication in this case are method arguments or a serialized version of the contract in XML format [40][11], which can also be wrapped in XML-based documents such as SOAP messages.
- *Adaptor connectors* are required to transform messages—often XML-based—into an object method call or any other kind of procedure call of the message receiver's implementation language, and from this language to a message format that can be understood by the message recipient.
- *Event connectors* are used to notify negotiation participants about the negotiation status. MAGNET, for example, holds a *participant* running at the auction server on behalf of an agent, which notifies its remote agent of new events within the auction. In ContractBot—most specifically in AuctionBot—events are also notified to subscribed parties via e-mail. ECBNS uses a domain specific language—namely the Formal Language for Business Communications (FLBC)—to allow application parties to communicate through meaningful business-termed messages.
- *Data access connectors* provide persistence of negotiation objects (MAGNET) or provide access to a product/service offerings database (COSMOS). ContractBot uses as well a data access connector to load auction parameters and current bids from a database.
- *Arbitrator connectors* mediate resource and state access required by multiple components, such as in multi-threaded systems that required shared memory synchronization. Little-JIL for example, communication of concurrent processes or "steps" is achieved via shared memory access.
- In some cases complex *custom connectors* support both internal and external communication. For example, ERA's architecture includes a Secure Messenger connector which mediates interaction with external parties. This connector also provides message encryption, authentication, verification, and translation services. The HP Contract Framework builds a Contract Network Protocol connector on top of SOAP/HTTP. Other multi-tiered architectures include middleware to handle message distribution mechanisms [53] or to make distributed object calls with CORBA [40].

Table 4. Negotiation - communication properties

	REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
	ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
Connectors	event	Secure Messenger / PC	n/a	n/s	RPC + distributor	RPC / adaptor	RMI / event / adaptor/ data access	n/a	n/a	arbitrator
Data format	FLBC	messages / method arguments	n/a	n/s	method arguments	serialized object or XML contract	method arguments / messages / objects	n/a	n/a	shared memory

ContractBot	HP	E-ADOME	eMediator	ER^{EC}
distributor / data access	RPC / adaptor	n/a	PC or RMI / data access	n/a
method arguments	SOAP	n/a	method arguments	n/a

Assessing the communication properties—software connectors and communication formats—was challenging in many cases given the unavailability of source code, unclear descriptions, and diverging terminology. Furthermore, many architectures involved high level descriptions which lacked the depth to which specific communication mechanisms were discernible [77][17]. These studies focused more on high level requirements of negotiation systems and their role within contracting architectures.

8.1.4 Distribution

We assess whether negotiation occurs in a distributed way, i.e. when parties communicate through a network with the intent of reaching an agreement, as opposed to whether the system's architecture is a distributed one.

The observed trend is for parties to negotiate distributedly through intelligent agents or through intermediaries, largely due to a matter of privacy and trust. Since negotiation often involves conflicting goals, parties try to conceal their negotiation strategies and private information. Therefore, the likelihood of a single party controlling the negotiation environment is low given the inconvenience, to say the least, for the counterparts.

- Most negotiation architectures rely on *intelligent agents* which are usually distributed [17] and communicate through technologies such as CORBA [53], TCP/IP interfaces [89], and RPC.
- Other *examples of distributed negotiation* are ERA, ContractBot, the HP Contract Framework, and Little-JIL. In ERA each party manages their own negotiation infrastructure and exchanges messages with other parties. ContractBot supports human negotiators through distributed Web clients. In the HP Contract Framework enterprises connect their contractual frameworks through a message-based communication bus. Little-JIL models negotiation distributed processes which assumes negotiation carried out by distributed agents.
- *Combined or hybrid approaches* are for example through mobile agents which can either execute in an agent dock within or near the auction server host computer to reduce negotiation latency [95]. Another example is MAGNET, which maintains proxies or *participants* within the negotiation server and their corresponding distributed agents. Participants notify agents on the auction status.
- In a few instances, negotiation participants rely on *trusted third parties* to mediate negotiation and host the negotiation environment. For example, COSMOS provides a *market* and brokering services to find commercial partners. However, it is not clear if negotiation takes place within or outside the market. Other examples are eMediator and ECNBS where negotiation management and support is hosted within a trusted third party site.

Table 5. Negotiation - distribution

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/s	distributed	n/s	distributed agents	distributed agents	distributed market participants	centralized proxies / distributed agents	n/a	n/a	distributed processes (negotiation)
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				distributed web clients or agents	distributed deployment	n/a	distributed or non-distributed mobile agents	n/a	

8.1.5 Dynamic adaptation

With one exception, none of the evaluated systems support the dynamic adaptation of negotiation strategy nor negotiation location. The exception is ContractBot—and its underlying auction platform AuctionBot—where human negotiators can bid through web interfaces, thus their location can change without interrupting negotiations. This is however a very weak form of dynamic adaptation of negotiators location which is not prone to automation.

Although ERA does not support dynamic negotiation adaptation, it provides an evaluator component which retrospectively analyzes past contract relationships to adapt future contracting behavior.

Table 6. Negotiation - dynamic adaptation

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				humans bid from any web interface	n/a	n/a	feasible through agent mobility	n/a	

8.1.6 Automation

In our evaluation, we found studies which either partially or fully support negotiation automation.

- COSMOS enables *semi-automatic* collaborative contract establishment through the exchange of offers and counteroffers.
- AEW supports *automated negotiation* of work distribution through intelligent agents. Also, in both MAGNET and eMediator agents semi-automatically or automatically bid within domain specific markets. eMediator provides an in-depth description of the diverse negotiation capabilities and strategies of its supported mobile agents. Although ContractBot delegates negotiation to the AuctionBot auction platform, it enables the automatic generation and configuration of auctions through different combinations of negotiation parameters. Automated negotiation can be achieved through the specification of a negotiation process the Little-JIL executable process language.
- Although some studies are silent on automation, their architecture and mechanism descriptions can *implicitly suggest automation support* [89]. For example, Cheung et. al derive negotiation plans which suggest will be automatically followed through by an agent. However, it is not clear if the plan is derived with the intention of simply supporting the human negotiation process or enabling negotiation automation.

Assessing the extent to which automation is supported is challenging when the mechanisms to achieve such property are not fully described or demonstrated [77][11]. ERA, for example, provides a reference architecture which leaves the automation mechanisms details to a specific application design.

Table 7. Negotiation - automation

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	n/s	n/a	n/s	automated work distribution negotiation	semi-automated collaborative contract formation	suggests semi-automated agent bidding	n/a	n/a	formal executable process language
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				automated auction creation	n/s	n/a	automated mobile agent bidding	n/a	

8.1.7 Security

In our evaluation framework there are two security perspectives with respect to negotiation: security from negotiation partners and communication security.

- With respect to business partners trust, the most common security provision observed within the analyzed studies is trusted third party negotiation mediation. This is the case for COSMOS, ECBNS, MAGNET, and eMediator. In these cases, an impartial party monitors and manages the negotiation infrastructure to discourage fraud and mediate conflicts. In addition, MAGNET requires party identification, enforces negotiation protocol rules, and implements party negotiation proxies to reinforce privacy. Both MAGNET and HP Contract Framework proactively track and store the negotiation state for auditing and conflict resolution. In Little-JIL, agents have limited knowledge about other agents and on how they make their decisions.
- With respect to communication security, many alternatives have been observed, such as digital signatures (public key systems) [40][6], message encryption [21][6], as well as verification and semantic message mapping [6]. In Cheung et. al a custom security layer was proposed but details are not provided. Lastly, eMediator provides a safe execution platform for negotiating mobile agents.

Table 8. Negotiation - security

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
3_{rd} party mediation	message encryption and verification / digital signatures	n/a	external comm. security layer	n/a	3_{rd} party mediation (broker) / digital signatures	mediation, identif. state tracking proxies, encryption	n/a	n/a	agents have limited knowledge of how others make decisions
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	n/a	3_{rd} party mediation / safe agent execution	n/a	

In the process of reviewing the literature we perceived that security—surprisingly—is not a central topic within contract negotiation. Some of the evaluated studies completely lack a discussion on security during negotiation [77][89]. Although some studies recognize that there are security issues which may require authorization, integrity, non-repudiation, privacy, authentication, digital signatures, and transport, messaging, and storage security, these are not directly addressed, but delegate the responsibility to the specific organization's business model implementing it [11].

8.2 Contracts

In the context of the contract phase we evaluate the number of parties involved in a contract (cardinality), if the contract language or contracting system allow specifying domain knowledge (scope), the concepts the contract model allows expressing, whether the contract can be dynamically adapted, if the contract model lend itself for automation, and whether exception handling clauses exist. The main findings in this phase are:

- Two party contracts are the most common type of contract cardinality.
- Contracts are moslty domain independent, but in some cases extensible to domain knowledge or domain specific.
- There are requirement-like models, and concrete semi-formal and formal languages contracts.
- The most common elements in a contract model are parties, roles, goods, price, obligations, permissions, and prohibitions.
- Dynamic adaptation of the contract is mostly not supported outside of the negotiation phase.
- Automation in this phase involves mostly contract formation.
- Contract templates are often the starting point.
- Some contract formalisms are more prone to automation than others.
- Most studies do not include exception handling clauses.

8.2.1 Interaction cardinality

In our evaluation, we found both types of contract cardinality: bi-party and multi-party contracts.

- The most common type of contract cardinality "in practice" involves *two parties* where one is the contractor or buyer, and the other is the supplier or seller. In most cases, bi-party contracts are the result of auctions, where many bidders participate, but only one is the winner (either to buy or to sell) [53][21][89][95]. Other studies which include bi-party contracts are [48] and [39]. Although Little-JIL does not produce an explicit contract model, the outcome of its supported negotiation models—one-to-one and one-to-many—is implicitly a bi-party contract.

- A more challenging contract specification is that which involves *multiple parties*. BCAF, COSMOS, and E-ADOME consider these complex interactions. However, neither Business Contract Framework nor COSMOS provide a concrete architecture or a working prototype to demonstrate its feasibility. E-ADOME proposes a practical approach by which multi-party contracts are just a set of bi-party agreements. However, multi-party contracts have been proposed at a conceptual level, but none of these studies have demonstrated their implementation feasibility.

Table 9. Contracts - interaction cardinality

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/s	n/s	multi-party	bi-party / multi-party	bi-party	multi-party	suggests bi-party	suggests bi-party	bi-party	bi-party
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				bi-party	n/s	multi-party as sets of bi-party	suggests bi-party	multi-party	

8.2.2 Scope

Every contract applies to a given domain, such as real state, internet services, and so on. The support for expressing concepts in a given domain can vary greatly. We assess if and how evaluated technologies—including a contract specification language—provide support for specifying contracts in a given domain.

In our evaluation, we observed three groups of technologies with respect to contract scope: domain independent, extensible, and domain specific.

- Domain independent are those contracts which do not focus on any particular domain, but provide standard clauses and variables. In this category fit studies which refer to contracts in general terms.
- In the second group we found technologies that are not associated with any particular domain, but support augmenting contracts with domain specific terminology and clauses. For example, in BCAF the contract domain is an important part of the domain model, and the contract explicitly specifies the domain and its business rules. ECBNS's architecture supports specialized niches by providing standard terminology to describe products in a particular domain. Cheung et. al grant capabilities to perform domain specific variations or escalations to standard contract templates. In E-ADOME a contract is a set of attributes and their values, which can be augmented with domain specific attributes. In ER^{EC} domain specific event-condition-action rules can be bound to the contract specification.
- In the third group are technologies that specifically address a particular domain. For example, In both AEW and Little-JIL the contract domain involves particular categories of work units to tasks to be distributed among processing components or agents. In MAGNET, the *market* architecture component provides ontologies to specify contracts in a particular domain. Lastly, BizTalk B2B focuses on intangible goods and services.

Table 10. Contracts - scope

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
leverages standard domain terminology	domain independent	domain independent	domain specific contract variations	categories of work	domain independent	domain specific (depends on market)	domain independent	intangible goods and services	tasks
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				domain independent	domain independent	additional domain specific attributes	domain independent	additional domain specific rules	

8.2.3 Expressiveness

There is considerable variation in the expressiveness of contract models and it is greatly influenced by the application domain and the concepts and rules that need to be captured. A contract model can be as restricted—e.g. providing only product or service description, quantity, and price—or as expressive— including other clauses and terminology—as needed.

The most common elements within a contract model—present in all studies but one [53]—are the parties involved in the transaction, their roles (e.g. buyer and seller), description of the values to be exchanged, price, obligations, permissions, and prohibitions. AEW and Little-JIL are exceptions given that they are specific to task distribution, which includes the category of work, start and end times, work rate, and quality thresholds in its contract model.

Other elements in a contract model include contract validity period, schedules and deadlines, quantity, quality, currency, delivery and handling, domain and jurisdiction, activities, service enactment process description, refund and return policies, credit arrangements, distribution rights, customer service support, liabilities, exceptions, sanctions, decommitment clauses, and natural language descriptions.

The set of studies we evaluated included high level, requirement-like contract models [77], as well as concrete semi-formal and formal languages to describe contracts [39][48][11]. Contract models can be specified in a number of formats, such as XML-based documents—based on schemas and DTD—[40][39][48][63], as a collection of objects [40], as a set of attributes [19], as a set of normative statements (based on deontic logic operators)[11], or as declarative specifications in courteous logic programs [89].

Table 11. Contracts - expressiveness

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
objects, handling, delivery, refunds, credit, payments, distribution rights	n/s	roles, time period, goods description, obligations, quantity, frequency, quality, cost, domain	parties, obligations, permissions, prohibitions, variables (e.g. rent, duration)	category of work, start and end times, work rate, quality threshold	who (parties and roles), what (rights and obligations), how (steps and time), legal (terms and conditions)	n/s	concepts, activities, transitions, schedules, monitoring, payments, authentication, use, natural language description	parties, item, currency, jurisdiction, schedules, prices, delivery, contract interpretation, exceptions	task description, price, and delivery time/date.
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				goods, customer service, delivery, returns, restrictions of use, other terms and conditions	identities, roles, validity period, conditions, obligations, permissions, prohibitions, actions, deadlines	views, communication graphs between views, and attributes such as accept, offer, goal, schedule, payment, documents, QoS, exception rules, commit.	suggests buyer, seller, goods, price, decommitment clauses, deadlines or time of service	parties, roles, activities, rules, goods, payments, delivery, exceptions, subcontracts	

8.2.4 Dynamic adaptation

There is only one study which discusses dynamic adaptation of contracts. ERA allows run-time contract updates, which are stored separately to the *master contract* as "subsidiary arrangements". These updates are possible under required permissions and authorizations.

In a few cases, contract adaptations are done during contract negotiation, where contract templates are repeatedly modified and exchanged between negotiating parties [40][11]. However, this adaptation is part of the negotiation process and does not fit in our evaluation criterion, which is interested in contract adaptation during contract execution stage as a mechanism for handling exceptions.

The AEW study asserts that all parties need to consent any contract changes, but does not address whether these changes can be done at run time.

Table 12. Contracts - dynamic adaptation

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	contract data updates or subsidiary arrangements	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	n/a	n/a	n/a	

8.2.5 Automation

Within the selected studies, we have observed various levels of automation support for contracts, and in a few cases complete absence of such support [84][18][6]. Contract formation can be automatically or semi-automatically through contract templates. Specification languages can also enable automation.

- *Automated contract establishment* is the most relevant and challenging type of automation. Support for contract formation is either semi-automated or automated. Semi-automation of contract formation requires human input or decision making. Examples are COSMOS, BizTalk contracts—assembled by humans from standard clauses—and Cheung et. al. Examples of automated contract establishment are ContractBot, AEW, CrossFlow, Little-JIL, and possibly MAGNET.
- Often contract formation begins with a pre-existing *contract template* with standard re-usable clauses, whose values are negotiated or decided upon [40][48]. Some argue that a contract model

should identify only those semantically meaningful contracts parts which enable automation [40]. For example, ContractBot's contracts are a function of a contract template and the auction results.

- The choice of language to describe electronic contacts is tightly related to whether such contract is prone to automated execution or provides the opportunity to implement some automated capabilities to support the execution process. For example, ContractBot produces an executable contract representation—namely a courteous logic program—that can be used by an underlying workflow management engine. COSMOS, BizTalk Contracts, CrossFlow, the HP Framework, and ER^{EC} represent and exchange contracts as XML documents which promote some degree of automated processing. Alternatively, COSMOS provides an object oriented representation of the contract which can be directly manipulated by backend systems.
- *Other types of automation support* found were automated contract signing and approval [48], contract optimization (based on payoffs maximization) [95], as well as contract validation and reactive contract enforcement [77].

Table 13. Contracts - automation

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	n/s	contract legal validation / reactive contract enforcement	semi- and possibly full contract formation automation	automated contract formation	semi-automated contract formation	suggest automated contract formation	automated decision-making for contract formation	semi-automated contract creation / automated approval / signing and monitoring	automated contract establishment
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				automated auction and executable contract creation	XML contract model	n/a	automated contract optimization	XML contract model	

8.2.6 Exception handling

Despite that clauses for dealing with contract non-compliance and unexpected situations are fundamental in a business contract, most of our evaluated systems do not include their specification within the contract model. Exceptions are COSMOS, BizTalk B2B, E-ADOME, eMediator, and ER^{EC}.

COSMOS contracts describe how to deal with parties that do not comply with their obligations on time. eMediator's *leveled commitment contracts* assume upfront the possibility that a party might withdraw their participation in a contract and predefines penalties to pay in such a case.

BizTalk Contracts, E-ADOME, and ER^{EC} take a rule-based approach to exception handling. BizTalk specifies policy rules of prohibition and the legal jurisdiction under which conflicts are resolved. E-ADOME and ER^{EC} specify exception rules as a set of event-condition-action statements which describe contingency actions or legal consequences.

Table 14. Contracts - exception handling

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	n/s	n/a	n/a	n/a	exceptions are part of the contract model	n/s	n/a	specification of contingency rules and legal jurisdiction for conflict resolution	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	exception rules in contract as event-condition-action rules	leveled commitment contracts with decommitment penalties	contract exceptions as event-condition-action rules	

8.3 Workflow

In this phase, we will assess whether a single or multiple organizations participate in the workflow (cardinality), whether the WfMS is specific to a particular domain, what are the exhibited connectors and exchanged data formats, whether work units are distributed, if the workflow is inter-organizational (decentralization), if the workflow can be dynamically adapted, what are the security measures in this stage, how parties are accountable for their activities, and if workflow exceptions are handled. The main findings in this phase are:

- All of the evaluated systems with a workflow component support inter-organizational interactions.
- Most of the evaluated workflow management systems are domain independent.
- The most common connectors—both within and among organizations—are event connectors, remote procedure calls, and adaptor connectors.
- Exchanged data is most often in an XML-based format.
- All surveyed technologies support distributed interaction and activity execution.
- All evaluated systems which include a workflow component are decentralized.
- The most common reason for dynamic workflow adaptation is exception handling, followed by adaptation to new requirements.
- Although some portions of the workflow are automated or semi-automated, full automation has not been realized within any of the evaluated studies.
- Automation and dynamic adaptation are tightly related since adaptations may occur automatically.
- Automated exception handling is contingent upon whether exceptions are expected and their nature is known a priori.
- Trusted third parties audit, monitor, and control the contracting process between organizations.
- Other security mechanisms are logging, workflow views, incremental delivery of goods and payments, and custom security components.
- Accountability is achieved though trusted third party monitoring and direct inquiry, as well as message non-repudiation, event logging, event subscriptions, and workflow views.
- There are many mechanisms to deal with exceptions, as long as those exceptions are expected.

8.3.1 Interaction cardinality

One of the goals of B2B technologies is achieving integration and interoperation of systems within multiple organizations for the coordination and optimization of business processes. Every evaluated reference supports—either at a conceptual or implementation level—inter-organizational collaboration, communication (through application interfaces or web services), and workflow linkage mechanisms.

8.3.2 Scope

A workflow is carried out by a composition of both domain independent and domain specific components. For example, although AEW is not bound to any particular domain, it supports domain

Table 15. Workflow - interaction cardinality

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
inter-org. and intra-org.	inter-org.	inter-org.	inter-org.	inter-org.	inter-org.	inter-org.	inter-org.	inter-org.	inter-org.
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	suggests inter-org.	inter-org.	inter-org.	inter-org. and intra-org.	

specific tasks. We assess whether workflow technologies within the evaluated studies support enacting contracts in any domain, or if they are specialized for a particular one.

The workflow infrastructure of most of the surveyed technologies is domain independent. eMediator is an exception which is more suited for products and services that can be divided into "chunks" for the safe exchange through incremental payments and deliveries without third party mediation. This is less costly and convenient when no physical goods need to be delivered.

An exception to how workflow is used in these studies is Cheung et. al's model, which leverages workflow management technology, not to enact the contract, but to support agent negotiation activities.

Table 16. Workflow - scope

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
domain independent	domain independent	domain independent	negotiation activities	domain independent	domain independent	n/s	domain independent	domain independent	domain independent
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	domain independent	domain independent	goods that can be divided in "chunks"	domain independent	

8.3.3 Communication properties

The most prominent connectors which mediate among workflow components—both within and among organizations—are event connectors and remote procedure calls. Other connectors are adaptor connectors and specialized custom connectors.

- In some cases, components are subscribed to some internal or external *event* [48][63], and at the event's notification, components execute some functionality or transfer data and control to other components through remote procedure calls. For example, BizTalk B2B includes a COM+ event subscription and notification services. In ER^{EC} the "publish" and the "subscribe" web services are used to notify business partners of new events and to register for an event subscription and a specific method of notification delivery correspondingly.

- Many architectures adopt an object oriented approach, leveraging *Remote Method Invocation (RMI)* to transfer data and control. For example, the COSMOS workflow engine calls *business objects*—representing people, products, departments, and so on—through RMI and method arguments. E-ADOME leverages object oriented mechanisms to match agents to workflow activities. ECBNS is based on an extensible object framework where processes are embedded within objects.

- Another observed connector type is the *adaptor connector* which enables communication between components not designed to cooperate. For example, BizTalk uses MSMQ triggers to associate messages in a queue with message-handling COM components, as well as wrappers to communicate between business components and the message routing infrastructure. In ERA, adaptor connectors—*external* and *internal mappers*—provide translation services for incompatible organization-specific message schemas, and between a contracting manager and process enactment components. CrossFlow provides *proxy-gateways* to mediate cross-boundary communication and perform syntactical

translations. All XML-based messages require adaptors to translate them to a format understood by backend components [11] or match them to the correspondent web services [18].

- We found a single instance of the *data access connector* in BizTalk, where contract values are stored in a database.
- *Arbitrator connectors* were found in Little-JIL, where communication of concurrent processes or "steps" is achieved via shared memory access.
- Some of the evaluated architectures included *custom complex connectors* that fulfilled specific communication and security requirements. For example, in AEW a custom connector enables intra-agent and inter-agent communication, between agents and workflow infrastructure, and between agents and end-users. In the HP Contract Framework, a custom *Contract Fulfillment Protocol* dictates communication rules—based on deontic operators—for agent communication. An *Internet Message Sender* in Cheung et. al sends ICQ alerts or e-mails to human agents and requests to agents' APIs, and an *Internet Event Interceptor* translates messages to ADOME events. In ERA, a *Secure Messenger* mediates communication among local e-contracting components and with business partners' contracting infrastructures, providing cryptographic, authentication, and message verification services.

With respect to the exchanged data format, messages are often in XML-based formats—such as BizTalk [48] or SOAP messages [11]—or EDI messages—as in ECBNS. For example, E-ADOME can invoke internal or external web services which return XML-formatted data. XML-RPC is also use to make remote procedure calls, where procedure name and arguments are specified in XML tags.

Table 17. Workflow - communication properties

	REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
	ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
Connectors	RMI / adaptor	Secure Messenger / PC / adaptor (integration mappers)	n/s	event (Message Sender) adaptor (Event Interceptor) RPC	custom communication module	RMI	n/s	RMI / adaptor (proxies)	event (pub/sub infrastructure) / adaptor (MSMQ Triggers) / data access	arbitrator
Data format	EDI messages	messages / method arguments	n/s	XML messages / procedure arguments	n/s	method arguments	n/s	method arguments	BizTalk Messages	shared memory
					ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
					n/a	RPC / adaptor	event / adaptor / RPC	n/s	event (pub/sub) / adaptor	
					n/a	SOAP messages	XML messages procedure arguments	n/s	XML messages	

8.3.4 Distribution

In our evaluation we found that all surveyed technologies support distributed interaction and activity execution. For example, BCAF is designed for open distributed systems. In AEW and E-ADOME workflow is supported by a community of distributed and autonomous agents. COSMOS describes workflows through distributed Petri Nets. In eMediator, exchange is achieved through incremental distributed service or product deliveries. In ER^{EC}, contract execution is driven by distributed event-condition-action processing. ECBNS supports distributed transactions and business process sharing. The Little-JIL process language is applicable to the modeling distributed processes.

Software connectors and implementation technology are indicators of whether a system is distributed. For example, AEW includes a CORBA-compliant platform, CrossFlow communicates through RMI middleware, and BizTalk implements COM+ components to support distributed transactions.

Furthermore when workflow involves multiple organizations, each organization is in control of its local workflow, which are linked together through messaging infrastructures [39][11][18][17].

However, in some cases workflow management is driven by the contracting workflow. For example, in AEW a central administration distributes work among a set of work processing centers. Another example is COSMOS and CrossFlow where business processes are outsourced and components from other organizations are included within local workflows.

Table 18. Workflow - distribution

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
distributed transactional workflow	suggests distributed	open distributed systems	distributed	distributed software agents and processing units	distributed Petri Net workflow implementation	suggests distributed	distributed outsourced business process	suggests distributed	specification of distributed processes
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	distributed	distributed agents	suggests distributed	distributed	

8.3.5 Decentralization

All evaluated systems which include a workflow component are decentralized, where autonomous parties negotiate, reach agreements, and collaborate through combined business processes. Decentralization is inherent when there are negotiation and contracting components that mediate interests of individuals and organizations, and when workflows span organizational boundaries. Systems involving a combination of business process from multiple organizations are *virtual enterprises* [39].

Table 19. Workflow - decentralization

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
possibly decentralized	decentralized	autonomous contract parties	decentralized	autonomous software agents and processing units	decentralized business objects	yes - contractor and suppliers	yes - virtual enterprises	decentralized	decentralized agents
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	autonomous contract parties	autonomous business partners	yes - buyers and sellers	decentralized	

8.3.6 Dynamic adaptation

Workflows are dynamically reconfigured due to changing organization or system requirements, or to cope with errors and exceptions such as unavailable services or non-compliant business partners.

- *New requirements* require workflow reconfiguration. In AEW, workflow configurations are contingent on new tasks and current processing capabilities of *work processing centers*. Exception handlers can be added, deleted, or modified at run-time in E-ADOME. Also, extending workflows or dynamically composing new ones is proposed by ER^{EC} to meet business agreements during contract execution. ECNBS proposes cloning workflows by downloading remote workflow scripts and instantiating them locally for just-in-time execution. Finally, CrossFlow supports the dynamic setup of the service enactment architecture and the dynamic linking of workflow components.
- We found that the most common reason for dynamic workflow adaptation is *exception handling*. For example, AEW supports real-time exception handling and redistribution of unprocessed work. In BCAF contract enforcement is reactive to current events, triggering corrective actions in case of

contract violation. Contingency and compensation actions suggest workflow adaptation in ECNS. In MAGNET, the contractor is continuously monitoring and repairing its service outsourcing plan. E-ADOME automatically resolves expected exceptions for workflow recovery. ER^{EC} supports transaction rollback and workflow re-generation. CrossFlow allows specifying alternative workflow execution paths and the circumstances under which those paths are executed.

Table 20. Workflow - dynamic adaptation

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
possibly through contingency actions and just-in-time execution of workflow scripts	n/s	reactive contract enforcement	n/a	real-time exception handling and work redistribution	n/s	suggests adaptation through outsourcing plan monitoring and repairing	dynamic service outsourcing and component linking, and alternative execution paths	n/a	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	dynamic workflow recovery and run-time addition/ modification of exception handlers	n/a	dynamically created workflows and adaptation to new requirements	

8.3.7 Automation

Workflow management endeavors include the definition, coordination, and automation of business processes. Despite this goal, full automation has not been realized within any of the evaluated studies. Many aspects of a business process still require human input, decision making, external knowledge, and semantic translations. However, automated or semi-automated activities exist in some portions of the business workflow. Automation depends on the application requirements [6].

Automation and dynamic adaptation relate and overlap, given that some adaptations take place automatically. We did not limit our evaluation to automation in the business process execution, but we assess as well whether any kind of automation exists during the contract enactment phase.

We observed that often exception handling activities are initiated and executed automatically. For example, financial penalties are imposed automatically when a contract violation has been detected in BCAF. AEW reactively handles exceptions by redistributing unprocessed work among other work processing units. E-ADOME takes a rule-based and event-based approach by explicitly specifying exception handlers in the form of event-condition-action, enabling automated resolution of exceptions and re-execution of failed activities. ECBNS executes automated procedures to undo the effects of committed actions in case of failure. ER^{EC} conducts exception-driven automated workflow recovery.

Automated exception handling is contingent upon whether exceptions are expected and their nature is known a priori. Automation is challenging when errors—both technical and within business activities—are unexpected and for which handlers have not been defined. In this case, alerts, notifications, and computer-based support to manual resolution can be provided like in E-ADOME.

Other forms of automation and semi-automation are concerned with business process execution [84][11][63], dynamic infrastructure setup (workflow generation) [39][63], response to new requirements such as work distribution [53], automated activity visibility [84], agent interaction [17], goods exchange plan generation [95], service remuneration [39], notifications and *business policy documents* [48], and the description and execution of automated negotiations through formal process modeling [14].

Many of the evaluated systems do not provide a workflow management and execution infrastructure, but are designed to work on top of any commercially available workflow management system, in which case, the level of automation support depends on the chosen WfMS. An example is AEW, where a layer

of autonomous agents is added on top of a WfMS to enable automated action in response to changes in the environment. COSMOS provides adapters to automated and manual workflow environments to execute the produced Petri Net-based specifications. E-ADOME, Cheung et. al, and ER^{EC} are built on top of ADOME WfMS, thus dependent on its automation capabilities.

Table 21. Workflow - automation

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
automation of cross-agency business processes, rollback, and activity visibility	n/s	automated contract enforcement (penalties)	suggests workflow technology-enabled automated agent interaction	automated work distribution and exception handling	n/a	n/s	automated workflow configuration, contract management, and payments	automated notifications and generation of policy documents	automated negotiation
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	automated execution of contract commitments	event-based automated expected exception handling	semi-automated exchange planner	enactment (semi-automated) exception handling, workflow creation, initiation, recovery	

8.3.8 Security

Security measures can be taken with respect to trust between business partners.

- The most common security measure against untrusted and unreliable business partners in the evaluated studies is relying on a *trusted third party* to audit, monitor, and control the contracting process between organizations [77][40][84][21][48]. This party maintains a copy of the contract and notifies when any party steps out of the contract boundaries in a way that could harm other parties. In BizTalk B2B Contracts, the trusted party generates business policy documents from the contract specification, which state the rules of behavior for each participating party. A certification authority may play the role of trusted party, by certifying and authenticating the authority and the competence of participating parties to fulfill a particular role, thus verifying the contract and contract parties [77].

- Also, logging mechanisms are leveraged to track activities and support non-repudiation among business parties [84]. In MAGNET, the *session* component ensures the continuity of transactions and prevents fraud through participant identification.

- Contract monitoring fosters trust among business partners and ensures the contract is carried out correctly, in a timely manner, and maintaining the agreed QoS [77][53][40][21][39][48][6]. For example, in ER^{EC} a *Contract Enactment Monitor* component tracks events in the workflow execution and infers violation according to activity-party-clauses and event-condition-action rules.

- E-ADOME's *"gray box interoperation"* is a different approach to privacy and security through *workflow views*, which allow business partners to expose only the relevant information to support workflow execution, balancing trust and security, and enabling inter-agency process monitoring. Similarly, Little-JIL assumes that collaborating agents have limited knowledge about each other.

- A security approach towards peer-to-peer trading is that of eMediator, which through the eExchangeHouse exchange planner enables the iterative, incremental, and alternating delivery of goods and payments to prevent contract non-compliance and minimize the possibility of loss.

Communication security depends on whether communication takes place within a private network or across the Internet [11]. Custom security components—such as CrossFlow's *proxy-gateways*, ERA's

secure messenger, and E-ADOME *access security layer*—protect organization boundaries by handling external communication, promoting information privacy and integrity. These components offer services such as message integrity check, encryption and decryption, digital signature management for sender's authentication, authorization, message verification, among others.

Table 22. Workflow - security

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
trusted 3_{rd} party control and log provides reliability	encryption, digital signatures, message integrity verification	certification and authority to fulfill a business role	E-ADOME's access security layer for external comm.	n/a	trusted 3_{rd} party controls marketplace and monitors contract execution	session ensures transaction continuity and fraud protection through identity verification	proxy-gateways control exit-entry to protect integrity and security	trusted 3_{rd} party monitors and enforces contract, and produces policy documents	agents have limited knowledge of each other
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	authorization, authentication, integrity, privacy, messaging and storage security, signatures	views protect private workflow details / authorization and authentication support	risk-mitigating incremental payment and delivery of goods	contract enactment monitor tests activities against clauses	

8.3.9 Accountability

Accountability can be achieved through trusted third party mediation, direct inquiry, message non-repudiation, event logging, event subscription, and workflow *views*.

- A *trusted third party* which monitors a contractual relationship not only provides some degree of security to business partners, but can provide an account of each party's activities [77][84]. Although some studies provide third party contract monitoring, it is not clear if and how parties could request such information [40][48].
- Parties can *request directly* from each other an account of the state of business activities [77][6], and the type of information that is permitted to be requested from business partners can be explicitly described in the contract as proposed by CrossFlow. In some cases only the contracting party is entitled to monitor and overview the state of its active contracts and outsourced services, such as AEW's *central administration* and MAGNET's contractor.
- *Message non-repudiation* means that parties cannot deny authorship of a message or service provenance [11][6]. The second is event logging—often maintained by a third party—where parties access log documents to have an account of their partners' activities, offering as well accountability of their own activities, existing in the upheld business relationship [84][39][63].
- *Event subscription* renders activities or service quality parameters visible to the authorized interested parties [84][39].
- E-ADOME enables accountability through *workflow views*, which expose the sufficient information from a private business process to provide an account of the contract status to business partners.

Table 23. Workflow - accountability

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
event log / non-repudiation / activity visibility	non-repudiation / ability to request info from counter-party	any party or trusted 3_{rd} party may monitor the contract	n/a	only contractor monitors contracts with providers	execution is monitored by the COSMOS provider	only contractor monitors outsourced services	online (observable events) and offline (log) monitoring	n/s	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	non-repudiation mechanisms	enabled by workflow views	n/a	possibly through event log	

8.3.10 Exception handling

Researchers argue that corrective actions to minimize deviations from the contract at the execution level—such as reactive enforcement or third party monitoring—are necessary when workflow activities abide by a contract [77][6]. We have found in the evaluated technology different mechanism to deal with exceptions at the workflow phase.

- In AEW agents collaborate in *real-time exception handling* by redistributing unprocessed work.
- ERA's reference architecture includes a *dispute handler* component to approach disagreements and support human monitoring and control.
- ECNBS relies on *pre-defined exception handlers, compensation transactions, and event logs*—which include exceptions—to restore process state in case of system crash.
- E-ADOME supports the *automatic resolution of expected exceptions* to recover workflow activities.
- Cheung et. al use E-ADOME to *intercept events* which might include exceptions.
- Crossflow performs transaction-like *compensation activities* to undo the effects of an activity.
- In MAGNET, contractors continuously monitor their outsourced activities and *create auctions to replace the uncompliant contractee.*
- ER^{EC}'s *activity-party-clauses*—linking parties, activities, and contract clauses—enable tracking violations and executing correspondent *exception handlers and compensation activities.*
- Lastly, *steps* or tasks within a Little-JIL's process model include exception handler specifications.

Table 24. Workflow - exception handling

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
workflow layer supports exception handling	dispute handler and enactment monitoring for human exception handling	specification of corrective actions	leverages E-ADOME exception handling capabilities	real-time exception handling and work redistribution	n/s	parties may decommit from a contract and pay a penalty	transactional compensation activities for rollback	n/a	*step* includes exception handlers
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	(expected) exception driven workflow recovery	n/a	violations tracking, exception handlers according to event-condition-action rules	

8.4 Architectural styles

An architectural style encompasses a set of design decisions that are considered appropriate for a given application context [106]. We assessed whether the surveyed technologies are explicitly designed following the principles of a known architectural style or whether their configuration and communication properties suggest any particular style. We observed that only a third of the evaluated technologies were designed with the explicit guidance of an architectural style's principles, being there a stronger emphasis on the implementation details and the technologies used.

Most of the evaluated architectures exhibit a combination of styles discernible at different levels of abstraction or existing in different places in the architecture. The predominant styles in the set of architectures are object oriented, distributed objects, event-based, and rule-based.

The *object oriented style* is based on components which encapsulate properties and methods to manipulate their values [106]. Architectures and domain models exhibiting object oriented qualities are ERA, Cheung et. al, HP Contract Framework, BizTalk B2B, E-ADOME, and eMediator. Although ERA is an event-based architecture at a higher abstraction level, sub-components provide methods that are invoked by higher level components. BizTalk relies on COM+ object oriented components to implement the business logic required to execute contract obligations. E-ADOME—and consequentially Cheung et. al's model which proposes implementation on top of E-ADOME—has an object-oriented capability to match agents with corresponding workflow tasks. Both the HP Framework and eMediator are implemented with object-oriented technology (i.e. Java), therefore we can freely infer object oriented communication behavior. eMediator's mobile agents are designed as objects.

The *distributed objects* is a composite style based on the object oriented and the client/server style and which leverages remote procedure call connectors for object communication [106]. ECBNS, AEW, COSMOS, MAGNET, and CrossFlow explicitly endorse or implicitly suggest an architecture based on distributed and potentially heterogeneous objects. ECBNS's architecture is based on distributed objects workflow technology that enables sharing business processes. Agents within the AEW are implemented as objects which communicate through a CORBA-compliant distributed computing platform. COSMOS is based on the CORBA Business Objects Architecture, a distributed object-oriented approach which allows deploying components remotely at runtime. MAGNET includes an auction server built as a distributed Java implementation, as well as client APIs which communicate through remote procedure calls with the server. CrossFlow also suggest a distributed object architecture given their Java implementation and the use of RMI. Finally, Juliette—an interpreter for Little-JIL's processes—is built on distributed object substrate to be able to distribute pieces of the interpreter closer to the agents with which it interacts [15].

The *event-based* style is characterized by autonomous components communicating solely though asynchronous messages [106]. The architectures or domain models which exhibit event-based style properties are ECBNS, ERA, Cheung et. al, and E-ADOME. In ECBNS negotiation parties communicate through *formal language for business communications (FLBC)* messages. ERA suggests an event-based architecture given their notion of an *event data item* which is a notification of the occurrence of an event sent from one party to the other. Also, a *Secure Messenger* connector mediates communication with external parties as well as among internal architecture components. E-ADOME is an event-based inter-organizational workflow management system, where internal and external events trigger local workflow activities.

The *rule-based* style is a shared memory style which maintains a knowledge base of facts and rules and an inference engine which based on user input and the knowledge base attempts to resolve a given query [106]. Although none of the evaluated architectures match perfectly the rule-based profile, a few systems share some of the style's principles. For example, in ContractBot rules are generated in the negotiation process and included in the final contract, which is itself specified in a rule-based language. Also, contract templates or *partial contracts* include negotiation rules on preferences and constraints which are used to infer auction parameters for auction configuration. In E-ADOME, business parties can access a rule base to inform themselves of potential actions—such as exception handlers—taken by other parties upon certain events. Rules included in the contract are specified in the <Event, Condition,

Action> form and are the basis to handle inter-organizational messages. In ER^{EC} events and exceptions are handled through specified event-condition-action rules within an *ECA-rule Manager*.

At a high level of abstraction, most architectures propose a *client-server style* while only a few follow instead a less hierarchical peer-to-peer style. In the client-server style, autonomous remote clients request services from servers through a combination of network protocols and remote procedure calls [106]. AuctionBot—ContractBot's underlying auction platform—is a client-server system where humans and software agents bid, create auctions, and review their accounts on the auction server through web interfaces or agent APIs. The HP Framework offers as well a Web portal through which users interact with other contract participants. eMediator's eAuctionHouse is also a client-server auction platform.

In *peer-to-peer* architectures there is no clear-cut distinction between service consumer and service provider, but peers within a network can both service consume and provide services [106]. The HP Framework provides the flexibility of choosing between a client-server or a peer-to-peer configuration where peers communicate through messages, specifically SOAP messages over HTTP. eExchangeHouse, eMediator's safe exchange planner, produces an algorithm by which business peers exchange goods and services incrementally without third party mediation.

We also observed *publish-subscribe* architectures similar to event-based architectures, where a set of subscribers register to receive specific messages, and publishers broadcast messages to subscribers. In AEW a *central administration* agent invites *work processing centers* to bid on work tasks through multicast messages. The central administration can selectively exclude broadcasting to processing centers which have recently received work tasks or which have failed to deliver. BizTalk leverages an XML-based event subscription and notification service to notify interested parties about relevant events, for example notifying the accounting component when a bill has been received. In ER^{EC} events and data are communicated through a web-service-based publish-subscribe mechanism for distributed and decentralized environments.

In the *layered style* components are arranged into ordered layers, where layers above use services of layers below [106]. E-ADOME is a layered architecture where top level software and human agent interfaces interact with the lower level workflow management layer, and this uses bottom level object database and rule base. AEW is as well a three-tiered architecture which has a top layer of control and report GUIs, a middleware layer for information distribution, and a backend layer of agents and execution components.

The *mobile code style* was leveraged by eMediator, where code and state moves to be executed in another host [106]. In eMediator, mobile agents move to an agent dock closer or within the same host as the auction server to reduce negotiation latency when bidding or creating auctions on behalf of individuals and organizations. Given that these agents are autonomous, they can run remotely and participate actively in negotiation even when the user is not connected to the system. However, agents are implemented as objects, which inherently have a tight coupling quality.

Finally, the *blackboard style* which is a type of shared state style [106], allows concurrent *steps* or subprocesses to communicate through shared memory in Little-JIL.

Table 25. Architectural style

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
distributed objects / event based	suggests event based / object oriented	n/a	layered / rule-based / event-based / object oriented	layered / distributed objects / pub-sub	distributed objects	distributed objects	suggests distributed objects	pub-sub / suggests object oriented	blackboard / distributed objects
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				client-server / event-based / rule-based	peer-to-peer or client-server / object oriented	layered / rule-based / event-based / object oriented	client-server / object oriented / mobile code / peer-to-peer	pub-sub / rule-based	

8.5 Integration

In this subsection we evaluate whether any form of integration exists between negotiation and contract, negotiation and workflow, and contracts and workflow. The main findings in this are:

- *Negotiation and contracts.*
  - A contract is the end result of negotiation.
  - A contract template can be input to the negotiation process.
  - Parties can collaboratively construct a contract.
- *Negotiation and workflow.* There is almost no direct integration of negotiation and workflow, but these phases are bridged through contracts.
- *Contracts and workflow.*
  - Contracts guide the course of the workflow.
  - Workflow descriptions or infrastructure configuration can be generated from contracts.
  - Contracts can also be executed by a workflow engine.

8.5.1 Integration of negotiation and contracts

Negotiation and contracts can be bonded through contracts as the negotiation result, though contract templates, collaborative contract proposals creation, among others.

- The *end result* to any successful negotiation process is a binding contract. A considerable amount of literature on negotiation focuses on the negotiation process and strategy and—surprisingly—do not address the resulting contract. Although the assumption is that the product of negotiation is an agreement between individuals and organizations, our evaluation approach—and this criterion in specific—considers only those studies which explicitly address contracts as the result of negotiation.

- Often the negotiation process starts with a *contract template* as input, which includes standard and reusable contract clauses—possibly domain-specific—and unbounded values which are negotiated and refined by humans or agents [77][53][17][11]. For example, ContractBot leverages an auction platform to fill out a contract template, generating a final contract with the auction results.

- In one-to-one or many-to-many negotiations, *contract proposals* are exchanged and collaboratively constructed or completed among the participating parties. In COSMOS, for example, each modification to the contract—to QoS values and business objects references in specific—is an offer which may be returned as a counteroffer or as a rejection. In the HP Framework, a *Contract Negotiation Protocol* allows modifying variables in contract drafts, as well as adding and removing contract clauses. In ERA, a *contractor* component creates offers and a finalized contract after an agreement has been reached.

- In addition, we found other ways in which negotiation and contract concerns are integrated and interplay. For example, Cheung et. al derive a negotiation plan based on the dependencies and priority of contract variables. eMediator instead proposes using two individual components in its tool suite—eCommitter and eAuctionHouse—to derive a contract and subsequently auctioning it.

Table 26. Workflow - integration of negotiation and contracts

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
final result of negotiation is a business contract	Contractor creates contract offers and ultimately a contract	contract template is input to negotiation / negotiation produces a final contract	negotiation refines contract template variables	agents negotiate to establish contracts	each contract modification (QoS attribute) is an offer or counteroffer	suggests negotiation result is a set of contracts with providers	n/a	n/a	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				contract generated from template and auction results	contract templates modified through Negotiation Protocol	n/a	eCommitter's contract can be auctioned with eAuctionHouse	n/a	

8.5.2 Integration of negotiation and workflow

The aftermath of a successful negotiation is a business process conducted by two or more parties. Although, these two components of e-commerce rarely interact directly—being mostly connected by a contract—there are cases where negotiation results directly impact the ongoing business process.

- In AEW the workflow adapts dynamically due to the auction results, distributing work tasks among processing units, and thus reconfiguring dynamically the workflow.
- Another example of how negotiation and workflow software components are integrated is through a shared underlying language for negotiation and workflow description, such as in ContractBot, where declarative courteous logic programs support both stages of e-commerce.
- In Cheung et. al's model, the E-ADOME agent-enhanced WfMS is leveraged for agent-based negotiation, where the negotiation plan is conducted as a workflow process. Similarly, negotiation processes are specified through the Little-JIL agent coordination language, which is a workflow technology. These are interesting examples of how workflow technology is leveraged to drive the negotiation process.

Table 27. Integration - negotiation and workflow

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/s	n/s	n/s	negotiation plan conducted as a workflow process	workflow reconfiguration due to negotiation results	n/s	n/s	n/a	n/a	negotiation driven by workflow technology / task assignment negotiation
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				common declarative language (courteous logic programs)	n/a	n/a	n/a	n/a	

8.5.3 Integration of contracts and workflow

Contracts between organizations—which are either consumers or providers—are the reason for the design and implementation of cross-organizational business workflows [39]. Contracts can relate and be integrated with workflows in different ways.

- Contracts guide and drive workflow activities. For example, in AEW contracts regulate the distribution and the flow of work, whereas in CrossFlow alternative workflows might be contingent on contract clauses.
- Contracts not only guide business workflow execution, but often explicitly describe the interactions for such execution. In ECBNS, for example, contracts are explicitly modeled as an agreement among distributed workflows, mapping compatible service use and provide interfaces. In E-ADOME e-contracts are modeled as communication graphs between workflow views which belong to different participating organizations.
- Another approach is the collaboration and interaction of software components that belong to either contracts or workflow. The *contract manager* and the *contract enactor* components in ERA communicate to enable contract execution.
- The contract can be the input to workflow components. For example, eMediator takes the contract reached through eAuctionHouse or eCommitter as input to *eExchangeHouse*, which derives a delivery and payment plan to follow. Another example is ECBNS where contracts in a contract base are input to the EDIFACT workflow manager which ultimately executes contracts.
- Contract artifacts can be mapped to workflow ones. For example, ER^{EC} maps the activities specified in the contract into a set of workflows, specifically XML contract descriptions to web services-based implementations. In the HP Framework, mappings are maintained between contract commitments and execution instances. At a obligation fulfillment request, a *fulfillment* component is guided by the commitments to determine which action to execute.
- A more automated approach is the contract-dependent generation of the enactment infrastructure—which connects dynamically service consumer and provider—as proposed by CrossFlow. Likewise, COSMOS includes execution definitions within the contract, which can be leveraged to generate Petri Net workflow representations. Also, contracts can be executable—for example as in ContractBot—and therefore interpreted by a workflow management system.
- Weaker forms of integration between contracts and workflow are for example using the resulting task assignment in MAGNET as a basis for defining execution schedules. BizTalk B2B generates policy documents from the contract—describing behaviors of participants—and from these derives business plan documents with detailed activities, which are ultimately mapped to executing business objects.

Table 28. Integration - contracts and workflow

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
contract base is input to the EDIFACT WfMS	Enactor collaborates with the Contracting Manager to execute the contract	n/s	contract defined in the negotiation workflow	contracts regulate the flow of work items	Petri Net-based workflow can be generated from the contract	task assignment is base for initial execution schedule	contract-based generation of execution infrastructure	contract → policy document → business plan document	implicit contract guides workflow
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				resulting contract is executable	mappings between contract and execution maintained	e-contracts include workflow views and communication graphs between views	eExchange House could carry out produced deal	contract descriptions mapped to web service-based workflows	

8.6 Bi-directionality

Following, we evaluate whether there exists the bi-directional flow of information and control between the different e-commerce phases. Our findings are:

- There is no bi-directionality between negotiation and contracts in the evaluated studies.
- Bi-directionality between negotiation and workflow mostly takes place in response to exceptions.
- Bi-directional interaction between contract and workflow components takes place mostly for information, guidance, monitoring, and enforcement purposes.

8.6.1 Bi-directionality between negotiation and contracts

In our evaluation of studies, we were unable to find examples of bi-directional relationship between contracts and negotiation, where parties went back to a negotiation phase to modify the current enacted contract. The only case is that of ERA, where the *contracting manager* is notified about a dispute during contract enactment for resolution. However, it is not clear whether the incumbent contract instance is the artifact subject to re-negotiation.

Table 29. Bi-directionality - negotiation and contracts

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	n/s	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	n/a	n/a	n/a	

8.6.2 Bi-directionality between negotiation and workflow

From our observations, going back from a contract enactment stage to negotiation mostly takes place in response to exceptions, such as contract non-compliance by any party. Examples are AEW, MAGNET, ERA, and Little-JIL. In AEW, processing centers notify a central administration of their inability to complete the assigned work, which is then re-auctioned and redistributed among other processing units. In MAGNET, the contractor repairs its outsourcing plan by calling out for bids when expected service delivery fails. Lastly, in ERA any party may request contract re-negotiation. Little-JIL's process descriptions can include *steps* that prescribe going back to a negotiation state in case, for example, that the winning bidder or contracted party fails to deliver the service promised.

Table 30. Bi-directionality - negotiation and workflow

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	re-negotiation to attempt to resolve disputes	n/a	n/a	agents can overbid to inform their current work capabilities	n/a	contractor repairs outsourcing plan by rebidding at failure	n/a	n/a	go back to negotiation *step* if specified
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	n/a	n/a	n/a	

8.6.3 Bi-directionality between contracts and workflow

Bi-directional interaction between contract and workflow components takes place mostly for information, guidance, monitoring, and enforcement purposes. We have observed examples where parties based their monitoring activities on the established contract to gain insight on ongoing business relationships—for example, contractors in AEW and MAGNET. The contract is leveraged not only to audit and monitor business activities, but to enforce them as well, taking corrective actions when activities step outside of the contract's boundaries [77].

Another observed case where the execution process refers and goes back to the contract stage is to modify the current contract upon parties' mutual agreement (e.g. ER^{EC}). These changes to the contract model and to the workflow itself can take place upon a service modification/update request from the consumer to the provider.

Table 31. Bi-directionality - contracts and workflow

REFERENCE ARCHIT.		DOMAIN MODELS		SYSTEMS					LANG.
ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	Little-JIL
n/a	n/a	through contract enforcement to ensure contract conformity	n/a	contractor monitors solution by referring to contracts	n/a	contractor continuously monitoring outsourcing plan	consumer may request provider to modify its process	n/a	n/a
				ContractBot	HP	E-ADOME	eMediator	ER^{EC}	
				n/a	n/a	n/a	n/a	support for dynamic modification of the contract model during execution	

8.7 Summary

In the following tables we provide a summarized but complete account of our evaluation of a representative set of architectures, domain models, and languages which integrate in some way negotiation, contracts, and workflow concerns.

Table 32. Evaluation ECBNS, ERA, BCAF, and Cheung et. al

		REFERENCE ARCHITECTURES		DOMAIN MODELS	
		ECBNS [84]	ERA [6]	BCAF [77]	Cheung et. al [17]
Negotiation	Cardinality	n/s	one-to-one / multiple issues	one-to-one / n/s	one-to-one / many-to-many / multiple issue
	Scope	standard exchange and payment terms	domain independent	domain independent	domain specific modifications to standard templates
	Communication	event (message) / FLBC	Secure Messenger / PC / messages / method args.	n/a	n/s
	Distribution	n/s	distributed	n/s	distributed agents
	Dynamic adaptation	n/a	n/a	n/a	n/a
	Automation	n/a	n/s	n/a	n/s
	Security	3_{rd} party mediation	message encryption and verification / digital signatures	n/a	external comm. security layer
Contracts	Cardinality	n/s	n/s	multi-party	bi-party / multi-party
	Scope	leverages standard domain terminology	domain independent	domain independent	domain specific contract variations
	Expressiveness	objects, handling, delivery, refunds, credit, payments, distribution rights	n/s	roles, time period, goods description, obligations, quantity, frequency, quality, cost, domain	parties, obligations, permissions, prohibitions, variables (e.g. rent, duration)
	Dynamic adaptation	n/a	contract data updates or "subsidiary arrangements"	n/a	n/a
	Automation	n/a	n/s	contract legal validation and reactive contract enforcement	semi- and possibly full contract establishment automation
	Exception handling	n/a	n/s	n/a	n/a
Workflow	Cardinality	inter-organizational and intra-organizational	inter-organizational	inter-organizational	inter-organizational
	Scope	domain independent	domain independent	domain independent	negotiation activities
	Communication	RMI / adaptor / EDI messages	Secure Messenger / PC / adaptor (integration mappers) / messages / method args.	n/s	event (Message Sender) / adaptor (Event Interceptor) / RPC / XML msgs. / procedure args.
	Distribution	distributed transactional workflow	suggests distributed	open distributed systems	distributed
	Decentralization	possibly decentralized	decentralized	autonomous contract parties	decentralized
	Dynamic adaptation	possibly through contingency actions and just-in-time execution of workflow scripts	n/s	reactive contract enforcement	n/a
	Automation	automation of cross-agency business processes, rollback, and activity visibility	n/s - depends on application requirements	automated contract enforcement (penalties)	suggest workflow technology-enabled automated agent interaction
	Security	trusted 3_{rd} party control and log provides reliability	encryption / digital signatures / message integrity verification	certification and authority to fulfill a business role	E-ADOME's *access security layer* for external comm.
	Accountability	event log / non-repudiation / activity visibility	non-repudiation / direct info request to counter-party	any party or trusted 3_{rd} party may monitor the contract	n/a
	Exception handling	workflow layer supports exception handling	dispute handler and enactment monitoring for human exception handling	specification of corrective actions	leverages E-ADOME exception handling capabilities
	Architectural style	distributed objects / event based	suggests event based / object oriented	n/a	layered / rule-based / event-based / object oriented
Integration	Negotiation and contracts	final result of negotiation is a business contract	*Contractor* creates contract offers and ultimately a contract	contract template is input to negotiation / negotiation produces a final contract	negotiation refines contract template variables
	Negotiation and workflow	n/s	n/s	n/s	negotiation plan conducted as a workflow process
	Contracts and workflow	*contract base* is input to the EDIFACT WfMS	*Enactor* collaborates with the *Contracting Manager* to execute the contract	n/s	contract defined in the negotiation workflow
Bi-directional.	Negotiation and contracts	n/a	n/s	n/a	n/a
	Negotiation and workflow	n/a	re-negotiation to attempt to resolve disputes	n/a	n/a
	Contracts and workflow	n/a	n/a	through contract enforcement to ensure contract conformity	n/a

Table 33. Evaluation AEW, COSMOS, MAGNET, and CrossFlow

		SYSTEMS			
		AEW [53]	COSMOS [40]	MAGNET [21]	CrossFlow [39]
Negotiation	Cardinality	one-to-many / single issue	one-to-one and possibly many-to-many / multiple issues	one-to-many / n/s	n/a
	Scope	work items bidding	offering's QoS attributes	domain ontology provided by a *market* component	n/a
	Communication	RPC + distributor (multicast) / method arguments	RPC / adaptor / serialized object or XML contract	RPC (RMI) event / adaptor / data access / method args. / msgs. / objects	n/a
	Distribution	distributed agents	distributed market participants	centralized proxies distributed agents	n/a
	Dynamic adaptation	n/a	n/a	n/a	n/a
	Automation	automated work distribution negotiation	semi-automated collaborative contract formation	suggests semi-automated agent bidding	n/a
	Security	n/a	3_{rd} party mediation (broker) digital signatures	mediation, identif. state tracking proxies, encryption	n/a
Contracts	Cardinality	bi-party	multi-party	suggests bi-party	suggests bi-party
	Scope	specific to categories of work	domain independent	domain specific (depends on market)	domain independent
	Expressiveness	category of work, start and end times, work rate, quality threshold	who (parties and roles), what (rights and obligations), how (steps and time), legal (terms and conditions)	n/s	concepts, activities, transitions, schedules, monitoring, payments, authentication, use, natural language description
	Dynamic adaptation	n/a	n/a	n/a	n/a
	Automation	automated contract establishment	semi-automated collaborative contract formation / object or XML contract model	suggest automated contract establishment	automated decision-making for contract establishment / XML + DTD contract model
	Exception handling	n/a	exceptions are part of the contract model	n/s	n/a
Workflow	Cardinality	inter-organizational	inter-organizational	inter-organizational	inter-organizational
	Scope	domain independent	domain independent	n/s	domain independent
	Communication	custom comm. module / n/s	RMI / method arguments	n/s	RMI / adaptor (proxies) / method arguments
	Distribution	distributed software agents and processing units	distributed Petri Net workflow implementation	suggests distributed	distributed outsourced business process
	Decentralization	autonomous software agents and processing units	decentralized business objects	yes - contractor and suppliers	yes - virtual enterprises
	Dynamic adaptation	real-time exception handling and work redistribution	n/s	suggests adaptation through outsourcing plan monitoring and repairing	dynamic service outsourcing and component linking, and alternative execution paths
	Automation	automated work distribution and exception handling	n/a - dependent on the underlying WfMS	n/s	automated workflow configuration, contract management, and remuneration
	Security	n/a	trusted 3_{rd} party controls marketplace and monitors contract execution	*session* ensures transaction continuity and fraud protection through identity verification	*proxy-gateways* control exit-entry to protect integrity and security
	Accountability	only contractor monitors contracts with providers	execution is monitored by the COSMOS provider	only contractor monitors outsourced services	online (observable events) and offline (log) monitoring
	Exception handling	real-time exception handling and work redistribution	n/s	parties may decommitt from a contract and pay a penalty	transactional compensation activities for rollback
	Architectural style	layered / distributed objects / pub-sub	distributed objects	distributed objects	suggests distributed objects
Integration	Negotiation and contracts	agents negotiate to establish contracts	each contract modification (QoS attribute) is an offer or counteroffer	suggests negotiation result is a set of contracts with providers	n/a
	Negotiation and workflow	workflow reconfiguration due to negotiation results	n/s	n/s	n/a
	Contracts and workflow	contracts regulate the flow of work items	Petri Net-based workflow can be generated from the contract	task assignment is base for initial execution schedule	contract-based generation of execution infrastructure
Bi-directional.	Negotiation and contracts	n/a	n/a	n/a	n/a
	Negotiation and workflow	agents can overbid to inform their current work capabilities	n/a	contractor repairs outsourcing plan by rebidding at failure	n/a
	Contracts and workflow	contractor monitors solution by referring to contracts	n/a	contractor continuously monitoring outsourcing plan	consumer may request provider to modify its process

Table 34. Evaluation BizTalk B2B Contracts, ContractBot, HP Contract Framework, and E-ADOME

		SYSTEMS			
		BizTalk B2B Contracts [48]	ContractBot [89]	HP Contract Framework [11]	E-ADOME [18]
Negotiation	Cardinality	n/a	one-to-many / single issue	n/s / multiple issues	n/a
	Scope	n/a	auctions based on domain specific rules and attributes	general purpose - unbound contract template variables	n/a
	Communication	n/a	distributor / data access method arguments	RPC / adaptor / SOAP (CNP protocol)	n/a
	Distribution	n/a	distributed web clients or agents	distributed deployment	n/a
	Dynamic adaptation	n/a	humans bid from any web interface	n/a	n/a
	Automation	n/a	automated auction creation	n/s	n/a
	Security	n/a	n/a	n/a	n/a
Contracts	Cardinality	bi-party	bi-party	n/s	multi-party as sets of bi-party
	Scope	intangible goods and services	domain independent	domain independent	additional domain specific attributes
	Expressiveness	parties, item, currency, jurisdiction, schedules, prices, delivery, contract interpretation, exceptions	goods, customer service, delivery, returns, restrictions of use, other terms and conditions	identities, roles, validity period, conditions, obligations, permissions, prohibitions, actions, deadlines	views, communication graphs between views, and attributes such as accept, offer, goal, schedule, payment, documents, QoS, exception rules, commit, etc
	Dynamic adaptation	n/a	n/a	n/a	n/a
	Automation	semi-automated creation automated approval/signing automated monitoring / XML *B2BContract*	automated auction and executable contract creation / CLP contract model	n/s / XML contract model	ERRATA semi-automation (contract templates)
	Exception handling	specification of contingency rules and legal jurisdiction for conflict resolution	n/a	n/a	exception rules in contract as event-condition-action rules
Workflow	Cardinality	inter-organizational	n/a	suggests inter-organizational	inter-organizational
	Scope	domain independent	n/a	domain independent	domain independent
	Communication	event (pub/sub) / data access adaptor (MSMQ Triggers) / BizTalk Messages	n/a	RPC / adaptor / SOAP message (CFP protocol)	event / adaptor / RPC / XML msgs. / procedure args.
	Distribution	suggests distributed	n/a	distributed	distributed agents
	Decentralization	decentralized	n/a	autonomous contract parties	autonomous business partners
	Dynamic adaptation	n/a	n/a	n/a	dynamic workflow recovery and run-time addition/modification of exception handlers
	Automation	automated notifications and generation of policy documents	n/a	automated execution of contract commitments	event-based automated expected exception handling
	Security	trusted 3_{rd} party monitors and enforces contract, and produces policy documents	n/a	authorization, authentication, integrity, privacy, messaging and storage security, signatures	*views* protect private workflow details / authorization and authentication support
	Accountability	n/s	n/a	non-repudiation mechanisms	enabled by workflow *views*
	Exception handling	n/a	n/a	n/a	(expected) exception driven workflow recovery
	Architectural style	pub-sub / suggests object oriented	client-server / event-based / rule-based	peer-to-peer or client-server / object oriented	layered / rule-based / event-based / object oriented
Integration	Negotiation and contracts	n/a	contract generated from template and auction results	contract templates modified through *Negotiation Protocol*	n/a
	Negotiation and workflow	n/a	common declarative language (courteous logic programs)	n/a	n/a
	Contracts and workflow	contract → policy document → business plan document	resulting contract is executable	mappings between contract and execution maintained	e-contracts include workflow views and communication graphs between views
Bi-directional	Negotiation and contracts	n/a	n/a	n/a	n/a
	Negotiation and workflow	n/a	n/a	n/a	n/a
	Contracts and workflow	n/a	n/a	ERRATUM contract monitoring	n/a

Table 35. Evaluation eMediator, ER^{EC}, and Little-JIL

		SYSTEMS		LANGUAGES
		eMediator [95]	ER^{EC} [63]	Little-JIL [14]
Negotiation	Cardinality	one-to-many / many-to-many single issue	n/a	one-to-one / one-to-many n/s
	Scope	domain independent combinatorial auctions	n/a	domain specific agents and domain independent negotiation process
	Communication	PC or RPC (RMI) / data access method arguments	n/a	arbitrator shared memory
	Distribution	distributed or non-distributed mobile agents	n/a	distributed processes (negotiation)
	Dynamic adaptation	feasible through agent mobility	n/a	n/a
	Automation	automated mobile agent bidding	n/a	formal executable process language
	Security	3_{rd} party mediation / safe agent execution environment	n/a	agents have limited knowledge of how others make decisions
Contracts	Cardinality	suggests bi-party	multi-party	bi-party
	Scope	domain independent	additional domain specific rules	tasks
	Expressiveness	suggests buyer, seller, goods, price, decommitment clauses, deadlines or time of service	parties, roles, activities, rules, goods, payments, delivery, exceptions, subcontracts	task description, price, and delivery time/date
	Dynamic adaptation	n/a	n/a	n/a
	Automation	automated contract optimization	XML contract model	automated contract establishment
	Exception handling	leveled commitment contracts with decommitment penalties	contract exceptions as event-condition-action rules	n/a
Workflow	Cardinality	inter-organizational	inter-organizational and intra-organizational	inter-organizational
	Scope	goods that can be divided in "chunks" (e.g. data services)	domain independent	domain independent
	Communication	n/s	event (pub/sub) / adaptor XML messages	arbitrator shared memory
	Distribution	suggests distributed	distributed	specification of distributed processes
	Decentralization	autonomous buyers and sellers	decentralized	decentralized agents
	Dynamic adaptation	n/a	dynamically created workflows and adaptation to new requirements	n/a
	Automation	semi-automated exchange planner	enactment (semi-automated) exception handling, workflow creation, initiation, recovery	automated negotiation
	Security	risk-mitigating incremental payment and delivery of goods	*contract enactment monitor* tests activities against clauses	agents have limited knowledge of each other
	Accountability	n/a	possibly through event log	n/a
	Exception handling	n/a	violations tracking, exception handlers according to event-condition-action rules	*step* includes exception handlers
	Architectural style	client-server / object oriented / mobile code / peer-to-peer	pub-sub / rule-based	blackboard / distributed objects
Integration	Negotiation and contracts	eCommitter's contract can be auctioned with eAuctionHouse	n/a	n/a
	Negotiation and workflow	n/a	n/a	negotiation driven by workflow technology / task assignment negotiation
	Contracts and workflow	eExchangeHouse could carry out produced deal	contract descriptions mapped to web service-based workflows	implicit contract guides workflow
Bi-directional.	Negotiation and contracts	n/a	n/a	n/a
	Negotiation and workflow	n/a	n/a	go back to negotiation *step* if specified in the process model
	Contracts and workflow	n/a	support for dynamic modification of the contract model during execution	n/a

9 Discussion

Distribution of the studies

The evaluated studies involve agent, workflow management, business processing, contracting, negotiation, auction, and e-commerce systems. This suggests that there is considerable overlap of interests in these communities. While some of the evaluated studies span more evenly among negotiation, contracts, and workflow concerns, others emphasize one or the other. Figure 22 shows the distribution of the evaluated studies among these e-commerce concerns. To obtain this visualization we looked at the percentage of the criteria that a given study addressed.

Based on our observations, we found that no study addresses both negotiation and workflow without talking about contracts. Contracts are the bridging artifact between negotiation and business workflow phases, facilitating the transition among these e-commerce stages. In addition, workflow is overall better supported than negotiation and has stronger ties with the contract domain.

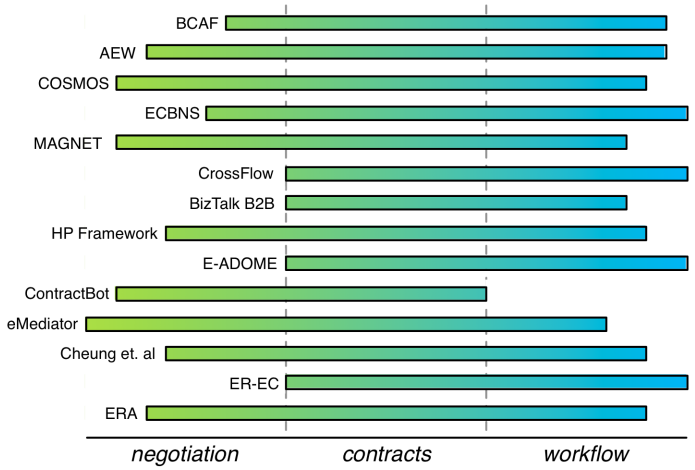

Fig. 22. Studies distribution

What has been prototyped

Given that our evaluation includes reference architectures, domain models, and languages not all the proposed functionality has actually been implemented, tested, and evaluated.

Table 36 shows the properties supported or addressed in the evaluated studies. A grey checkmark indicates that a property is addressed only at an architectural or conceptual level. A green checkmark indicates properties reported to be fully or partially implemented in prototypes. The green letter "E"—which stands for explicit—differentiates those architectures which are explicitly based on a known style or set of styles, from those which do not endorse styles explicitly labeled with a grey "~~E~~".

On this table we can observe that AEW and MAGNET exhibit most of the properties of interest, thus they have comparatively a stronger degree of integration of these concerns. Despite the existence of working and deployed systems that support either negotiation, contract management and specification, and business processing, this representative set of studies shows that there is still more work to be done with respect to the pragmatic integration and synergic interaction of these e-commerce components.

Based on this visualization, we consider that further research needs to be conducted to "fill the gaps" and achieve—at both architecture and implementation levels—the seamless integration between negotiation, contracts, and workflow phases and software components to more comprehensively support e-commerce among individuals and organizations.

Table 36. Visualization of implemented features

		REF. ARCH.		DOM. MOD.		SYSTEMS										LANG.
		ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	ContractBot	HP	E-ADOME	eMediator	ER-EC	Little-JIL
Negotiation	Cardinality		✓	✓	✓	●	●	●			●	●		●		●
	Scope	✓	✓	✓	✓	●	●	●			●	●		●		●
	Communication	✓	✓			●	●	●			●	●		●		●
	Distribution		✓		✓	●	●	●			●	●		●		●
	Dynamic adaptation										●			●		
	Automation					●	●	●			●			●		●
	Security	✓	✓		✓		●	●						●		●
Contracts	Cardinality			✓	✓	●	●	●	●	●	●		●	●	●	●
	Scope	✓	✓	✓	✓	●	●	●	●	●	●	●	●	●	●	●
	Expressiveness	✓		✓	✓	●	●		●	●	●	●	●	●	●	●
	Dynamic adaptation		✓												●	
	Automation			✓	✓	●	●	●	●	●	●	●		●	●	
	Exception handling						●	●		●			●	●	●	
Workflow	Cardinality	✓	✓	✓	✓	●	●	●	●	●		●	●	●	●	●
	Scope	✓	✓	✓	✓	●	●		●	●		●	●	●	●	●
	Communication	✓	✓		✓	●	●		●	●		●	●		●	●
	Distribution	✓	✓	✓	✓	●	●	●	●	●		●	●	●	●	●
	Decentralization	✓	✓	✓	✓	●	●	●	●	●		●	●	●	●	●
	Dynamic adaptation	✓		✓		●		●	●				●		●	
	Automation	✓		✓	✓	●	●		●	●		●	●	●	●	●
	Security	✓	✓	✓	✓		●	●	●	●		✓	●	●	●	●
	Accountability	✓	✓	✓		●	●	●	●			✓	●		●	
	Exception handling	✓	✓	✓	✓	●		●	●				●		●	●
	Architectural style	E	E		E	E	E	E	E	E	E	E	E	E	E	E
Integration	Negotiation and contracts	✓	✓	✓	✓	●	●	●			●	●		●		
	Negotiation and workflow				✓	●					●					●
	Contracts and workflow	✓	✓		✓	●	●	●	●	●	●	●	●	●	●	✓
Bi-directional	Negotiation and contracts															
	Negotiation and workflow		✓			●		●								●
	Contracts and workflow			✓		●		●	●						●	

On what is important

The goals, properties of relevance, and therefore the resulting architectures are different across the evaluated studies. Briefly, we describe what is important for each of the evaluated studies.

- *ECBNS*: supporting transactions in the context of distributed workflows.
- *ERA*: requirements and reference architecture for e-contract systems.
- *BCAF*: requirements and phases of cross-organizational contracting systems.
- *Cheung*: automated negotiation plan generation based on contract variables.
- *AEW*: automated process management with autonomous collaborating agents.
- *COSMOS*: broker and e-contracting service to support the contract lifecycle.
- *MAGNET*: negotiation and contract sessions in domain specific markets.
- *CrossFlow*: dynamic service outsourcing (search and infrastructure setup).
- *BizTalk B2B*: management of contracts, business policies, and plans.
- *ContractBot*: template-based auction generation and contract creation.
- *HP Contract Framework*: peer-to-peer support of contracting lifecycle.
- *E-ADOME*: interoperability and controlled visibility through workflow views.
- *eMediator*: combinatorial auctions, contract optimization, and safe exchange.
- ER^{EC}: contract modeling, enactment, and monitoring.
- *Little-JIL*: negotiation process modeling.

The emphasis and level of support of either negotiation, contracts, or workflow, and the relation among these components depends on the nature of the application domain and the design decisions considered appropriate for the specific problem that researchers and practitioners attempt to solve.

Commonality between these technologies

Although the evaluated technologies have different objectives as well a structural, behavioral, and technical properties, the commonality that binds them is the overarching goal of *facilitating, optimizing, and automating negotiation, contracting, and workflow processes* in one way or another. Technologies focus and overlap on finding business partners, planning and negotiating the terms of the agreement, establishing a final contract, and executing the terms of such contract. The difference among them is on how they achieve these goals. For example, they differ on the way that they find these potential business partner in the first place or on their negotiation style. The same applies for contract establishment, where they require different steps or algorithms and different automation proportions to achieve the same overarching goal.

Table 37. Areas addressed

	REF. ARCH.		DOM. MOD.		SYSTEMS										LANG.
	ECBNS	ERA	BCAF	Cheung	AEW	COSMOS	MAGNET	CrossFlow	BizTalk	ContractBot	HP	E-ADOME	eMediator	ER-EC	Little-JIL
Search	✔	✔	✘	✘	✘	✔	✔	✔	✘	✘	✘	✘	✘	✘	✘
Negotiation planning	✘	✘	✘	✔	✘	✘	✘	✘	✘	✔	✘	✘	✔	✘	✔
Negotiation	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✔	✘	✔	✘	✔
Contract establishment	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔
Contract enactment	✔	✔	✔	✘	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔

Cardinality

The dominating type of negotiation among the evaluated studies is one-to-many parties, specifically auctions where the initiator publicizes a service offering or request, and parties bid over the auction parameters—price for example—to either consume or provide. The result is a contract involving two parties. One hypothesis on why auctions prevail over other types of negotiation is because they are easier to implement given that there are only one or a few negotiable variables. Also, the auction creator has control over the conditions and restrictions of the contract, whereas in one-to-one negotiations control is distributed among parties, so the contract can change along many dimensions. Although two studies claimed support for many-to-many negotiation, one of them did not provide evidence of such implementation [17] and the other involves combinatorial double auctions [95]. Although Little-JIL does not describe many-to-many negotiation processes, nothing prevents the language from doing so. These complex descriptions can be useful to design and implement many-to-many negotiation systems.

Despite that negotiating multiple issues is more complex, it is mostly equally supported as single issue negotiation. Auctions usually negotiate over a single attribute (such as price), but in the evaluated studies two auction systems support bidding over multiple issues (at least at a conceptually) [40][17].

Contracts are predominantly bi-party. Despite that many studies refer to contracts as multi-party documents, we have not observed negotiations—excluding auctions—where multiple parties negotiate nor contracts—or contract templates—examples including multiple parties. A proposed workaround is to define multi-party contracts as a set of bi-party ones [18], or having subcontracts and composite contracts [63] as nested data structures to compose multi-party contracts. However, adopting the same approach for negotiations—that is composing multi-party negotiations from a set of bi-party negotiations—may be impractical at best since parties may have conflicting interests that need to be resolved in unanimity. Although all the surveyed technologies suggest that contract parties are organizations, we have not restricted our study to B2B e-commerce given that individuals are often auction participants. However, there are specific B2C e-commerce platforms such as Kasbah [16], which demand less user control and involve simpler processes.

Figure 23 shows a summary of each type of negotiation and contract cardinality by the number of studies. The green portion of the vertical bars indicate those systems which implement such negotiation or contract cardinality. The gray one indicates those which support the corresponding cardinality only at a conceptual level, as in the case of reference architectures and domain models.

With respect to workflow, all studies promote cross-organizational interaction given that they address as well contracts and negotiation, which inherently involve multiple self-interested parties.

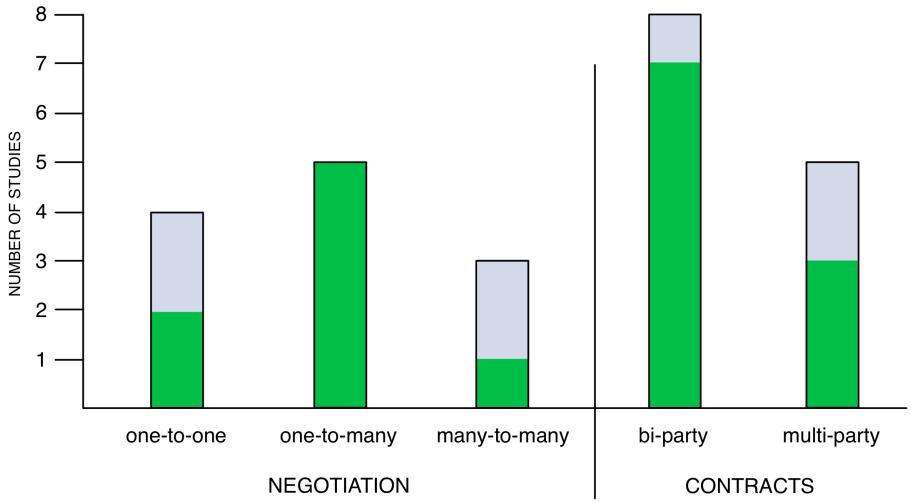

Fig. 23. Negotiation and contract cardinality

A shortcoming of the evaluated architectures, mostly of those coined as "e-contracting systems", is that they lack the flexibility and the customization capabilities to accommodate a particular business context. Namely they support specific types of interactions and business settings. Auctions are more flexible in terms of customization. For example eMediator allows different bidding facilities such as bidding via graphically drawn price-quantity graphs and through mobile agents, as well as enabling different auction configurations. However, auctions are a very specific kind of negotiation that involve mostly two-party contracts.

There is a need to promote flexible mechanisms that support different kinds of interactions, including more challenging ones where multiple parties have equal control over the negotiation and are able to modify the contract along various aspects and dimensions.

Negotiation support

In the set of studies we have evaluated, negotiation processes have different goals. These are, for example, distributing work among processing agents, outsourcing processes, buying or selling individual or combinations of goods and services, and so on.

Hence, evaluations—as the one conducted in this survey—are useful for application designers, since the assumptions of architectural frameworks, reference architectures, domain model, and systems about the negotiation context might not be suited for the task at hand.

We have observed that the support for negotiation is not as strong in these studies compared to workflow (figure 22). In addition, many studies—outside our selected ones—focus on negotiation mechanisms and strategies (e.g. cooperative vs competitive) [68], and despite it is assumed that the final product is a contract, oddly enough it is not explicitly stated.

A multi-agent system is the paradigm of choice for computer-based negotiation. Software agents act on behalf of humans or organizations, and are effective for evaluating and establishing profitable agreements and operating in complex settings, therefore saving negotiation time [94]. Agents are appropriate given their autonomy—and therefore decentralization—their goal-orientation, their reasoning and communication capabilities, and their self-interested nature comparable to human negotiators. We have not yet found in any of the researched literature an alternative to agents for negotiation.

A challenge during our evaluation was that "agent" is interchangeably used to refer to both software and human negotiators (through graphic user interfaces). Little-JIL is the only technology that makes a explicit distinction between human and software agents. There is a need to explicitly specify what is referred to by "agent" given that a system could support both human and software agents concurrently and the assumptions that can be made about automation capabilities.

The meaning of a contract

Despite the widely accepted definition of a contract as a binding agreement between two or more parties, we found that each study has a slightly different meaning depending on the application context. For example, CrossFlow contracts look at the interactions between agencies at a higher level of abstraction than the underlying workflow. In ECBNS a contract is a semantic agreement among collaborating distributed workflows. Other studies view contracts as monitoring artifacts [37] or as a process description [23].

In addition, *there are overlapping meanings where a contract is both a business document and the operational realization of these agreements through component interfaces.*

The specific meaning of contract depends on the application context and on the level of abstraction of the conversation.

Contract expressiveness and representation

Contract models greatly differ on their expressiveness, namely on the kind of concepts they are able to express. The least common denominator contract model across the evaluated studies includes parties and their roles, the nature of the exchanged good, the product or service price, and the period, frequency, time to deliver, or deadline. Other commonly supported concepts are obligations, permissions, and prohibitions, which are based on deontic operators and provide considerable expressive power to contract models given their open endedness.

In our evaluation, we found differences in the expressive power of contract models. An example of these—present in some models and absent in others—are product quantity, quality, currency, conditions and rules, exceptions and decommitments, return policies, handling and delivery method, credit, refunds, restrictions of use, customer service, steps and activities, distribution rights, legal jurisdiction, domain, monitoring specifications, contract validity period and use restrictions, and so on.

The contract data model heavily depends on the application domain and on the concepts that need to be expressed in a particular business context. A richer contract model is more expressive. However, expressiveness comes at a price, rendering the model more complex and hard to analyze and use. Also, depending on the application additional expressiveness might not be necessary.

Only ECBNS (limitedly), CrossFlow, BizTalk B2B, the HP Framework, E-ADOME, and ContractBot present a contract formalism, namely a concrete representation of the contract model. Most systems represent contracts as XML documents, which are passive documents—and therefore model passive contracts—which by themselves do not contribute to the dynamicity of a system, but are limited to the structured exchange of information. Only one study represented it as a set of objects and one in formal logic language (CLP). These representations provide different levels of automation. Objects and formal languages provide a higher degree of automatization. Furthermore, formal languages allow

identifying conflicting contract clauses. Two studies include as well a natural language description and contract interpretation guidelines to support human understanding [39][48].

With respect to the contract representation, structured and formal languages promote automation and unambiguity, however at the cost of understandability, simplicity, briefness of contract specification, and manipulability by non-experts.

Van der Aalst argues that expressiveness should take into account modeling effort, rendering an approach more or less suitable [108]. It is difficult to test and compare the expressiveness of the proposed formalisms given the limited design guidance and in most cases the description of simple and partial examples. Also, a detailed comparison is outside of the scope of this survey and more appropriate in a specialized discussion of contract specification languages.

Application designers need to analyze these tradeoffs and choose a contract representation language which suits specific requirements.

Contract templates

Partial contracts or contract templates include boilerplate clauses corresponding to particular kinds of agreements and unbound variables which are established upon negotiation. The contract template along with the agreed variable values constitute the final contract.

Negotiation in many of the evaluated studies starts off with a or a contract template [77][40][39][48] [89][11][17]. For example, repositories for standard contract forms and reusable contract clauses from which to create contracts are proposed in BizTalk B2B. Moreover, automated matchmaking of business partners is based on contract templates which include standard business communication protocols and service descriptions [39].

Discerning negotiated variables from fixed values largely depends on the business context and on what the negotiation initiator considers unconditional contract terms. For example, a consumer might need supplies delivered at a fixed date, deadline which is not subject to negotiation.

Researchers argue [40] that meaningful aspects of a contract which are amenable to computer manipulation and automation should be modeled, rather than attempt to capture the full complexity of a contract. The risk, however, is to lose important contract details in this process.

When modeling contracts it is important to find the balance between expressiveness and simplicity, analyzing what is the essential information that needs to be captured from natural language contracts and specifying the relevant obligations and business rules that will guide the cross-agency workflow.

While the benefits of contract templates are convenience, optimization, common understanding, and automation, the expressiveness and extensibility of custom made contracts is sacrificed.

Legal validity of contracts

The use of templates in e-contracting systems mimics the real world practice where services such as NOLO [1] provides "do-it-yourself" contract templates in a variety of legal domains. Standard machine readable contracts are a useful resource for application designers that require the certainty that e-commerce interactions are driven by valid and legally binding contracts.

Augmenting services such as NOLO by mapping natural language contract templates with equivalent machine readable contracts would promote standardization, allow creating custom contracts based on reusable legal boilerplate, and provide legal certainty and protection to parties.

In addition, legal guidance within e-commerce has prompted organizations such as the European Commission and its e-commerce directive to specify obligatory fields—thus promoting standardization—in B2B contracts to promote legal-based trust among business partners and protection to more vulnerable participants [5].

Only a few of the analyzed studies—COSMOS, ER^{EC}, and BizTalk B2B—discuss the legal validity of their contract models and the legal implications of contract violation in electronic activities. COSMOS associates parties or "legal entities" with signatures, and the contract model includes legal clauses associated with regulations and legislation. ER^{EC} and BizTalk B2B include in the contract document legal exceptions and the jurisdiction to which contract parties submit to resolve conflicts correspondingly. CrossFlow includes a natural language description of the contract where the legal context can be specified.

We have observed other mechanisms in which legal concerns are implicitly supported. For example, some studies [40][11][6] support digital signatures which certify the real-world identity of the parties to the contract, therefore making the contract legally binding and backed by the local laws of the contract parties. Also contract specification languages can bring a contract model closer to legal contract standards. For example, deontic logic—on which many contract languages are based [17]—is used to analyze and reason over the structure of normative law [76].

In addition, *learning from traditional contract structures and taking those as examples to build e-contract templates can provide more certainty that contract models are legally enforceable. Trusted intermediaries which mediate business interactions can also be legal instruments by providing legal support to monitor and enforce contracts, resolve conflicts, and impose penalties.*

Domain support

Every business interaction takes place in a domain. The complexity arises when contract templates need to accommodate different domains. Furthermore, contract templates might be unsuited for particular business situations and unconventional domains. This is the tradeoff between automation through contract templates versus the ability to customize contracts to specific needs.

Extensible contract templates could provide the required standardization and automation capability, while supporting custom requirements and domain concepts. Likewise, underlying contract specification languages need to provide the means to augment the baseline contract model to particular domain needs.

In the evaluated studies, we observed different means to support domain knowledge and mutual understanding for negotiation and contracting. One of these is supporting market specific ontologies, which are formal descriptions of the concepts in a given domain. Also, domain concepts can be embedded in the contract model. Rule-based systems—such as ContractBot—also provide the flexibility to include domain specific rules. Some systems are specifically tailored to a particular domain, so it is difficult to adapt them to other business context—as for example in AEW where agents negotiate work distribution. Despite some studies providing support to express domain knowledge, most studies are domain independent and do not provide any mechanisms to semantically describe domain concepts.

Given the extensive variations and different situations in business relations, e-commerce systems need to provide concrete support for specialized and meaningful interactions.

Glushko et. al [35] make a fair argument by stating that custom descriptions of business processes particular to a single or a small group of organizations will limit the opportunities for business relations, interoperability, and interaction with other organizations that might not share common models.

An alternative to support particular domain knowledge is to leverage established business standards.

For example, SweetDeal [42]—a system for contract representation—leverages standard process knowledge from the MIT Process Handbook. Other standards are RosettaNet—with a rich directory of "Partner Interface Processes"—cXML, xCBL, ebXML, OBI, among others. The challenge is that these business-oriented standards may not be suited for every situation and also they are mutually incompatible, thus fragmenting online markets in terms of readiness to communicate and interact.

Intermediaries

Third parties acting as intermediaries facilitate, audit, or monitor business partners interactions at both negotiation and contract enactment phases. As mentioned before, these third parties can be a legal entity, a financial institution, or other trusted party. Many of the evaluated studies assume the existence of third parties that mediate and manage the contracting infrastructure.

While the drawback of business intermediaries are increased transaction cost and escrow fees [95], the benefits are increased trust in the transaction, party accountability, and legal witnessing.

Also, third party mediators provide convenience in terms of the infrastructure. Take for example eBay where spontaneous negotiation takes place without having to setup an infrastructure. Although human negotiators are involved in this case, there are similar scenarios where digital negotiators leverage these intermediate infrastructures to find business partners and conduct business.

In contrast, eMediator demonstrates how intermediaries can be waived and replaced with peer-to-peer transactions. However, this mechanism which relies on incremental payments and deliveries is only valid for situations where intangibles are being exchanged or when the gain is worth the delivery cost. Therefore, in large business transactions this method might be too costly. Also the HP Framework relies on a peer-to-peer architecture, where each party holds its own contracting infrastructure.

Intermediaries are also specialized marketplaces or "B2B hubs" [3] which provide brokering services to search for goods. However, it is hard to engage with the same technology in many different hubs.

Although centralized markets provide a convenient infrastructure to find business partners, markets do not communicate or share information with each other, therefore causing market fragmentation instead of promoting market expansion [3].

Workflows

Workflows involve the management of information and the coordination of activities conducted by individuals and software to achieve some task. In the context of e-commerce, workflows constitute activities to provide a service or fulfill a variety of business contracts. Workflow systems face the challenge of supporting diverse communication channels, concurrent and dependent processes, and managing bottlenecks that may be caused by faulty software, human interaction dependencies, and third party processes.

E-commerce frameworks [35][100] largely involve workflow or their business-oriented successors—business process management systems—whose goal is the interoperability among cross-organizational business processes though communication technologies, protocols, and standards. Oddly enough, no system under the umbrella of "e-commerce frameworks" supports negotiation, even though it is an essential part of commerce.

Cheung et. al's domain model and Little-JIL are a distinguishing case on how workflow technology is used and how workflow concerns are approached. In Cheung et. al, an agent-enhanced workflow management system—i.e. E-ADOME—is proposed to implement a negotiation plan by leveraging E-ADOME's agent layer. Similarly, Little-JIL is used to explicitly describe negotiation plans or processes.

Negotiation is a specific type of workflow where parties take turns to execute some activity—such as making offers and counteroffers—so there is no reason why the same underlying technology could not be leveraged to both negotiate and execute a contract.

The way e-commerce components interact—negotiation, contracts, and workflow—is not uniform across studies and technologies. Flexible e-commerce infrastructures promote the creativity of application designers by allowing them to combine these technologies in innovative ways.

Communication

We assessed the kind of connectors mediating the interaction among negotiation, contracting, and workflow components, and the format of exchanged data. This assessment was challenging due to the unavailability of software artifacts and incomplete descriptions of architectural details in many cases.

The predominant type of connectors across the studies at both negotiation and workflow stages are remote procedure call—and remote method invocation—events, and adaptors.

Given that negotiation agents are often implemented as objects, their communication is based on method calls along with method arguments and serialized objects. In AEW, where agents are selectively invited to bid, a combination of RPC and distributor connectors achieve multicast communication.

Event connectors are leveraged—for example in MAGNET—where agents react to events of interest in the *market* and the *session* listens to events such as participant bids. Also, in workflow environments, some activities are triggered by events. Exchange data generally involves XML messages—often preferred for their human readability—and less often EDI messages (only within workflow). However, it is difficult to assess if these messages are only the medium to make remote procedure calls, i.e. XML-RPC. For example in HP Framework, messages are tied to the receiver's implementation details, being basically remote procedure calls embedded in XML messages. The drawback of both RPC and XML-RPC is introducing tight coupling among components and systems.

Adaptor connectors transform arriving messages—often XML-based—and translate them to a format understood by the receiver's implementation language.

Overall, we perceive a tight coupling between application components through remote procedure calls. In addition, all these technologies are described without an explicit rationale about the design decisions and the reflection of the properties the use of these connectors evoke.

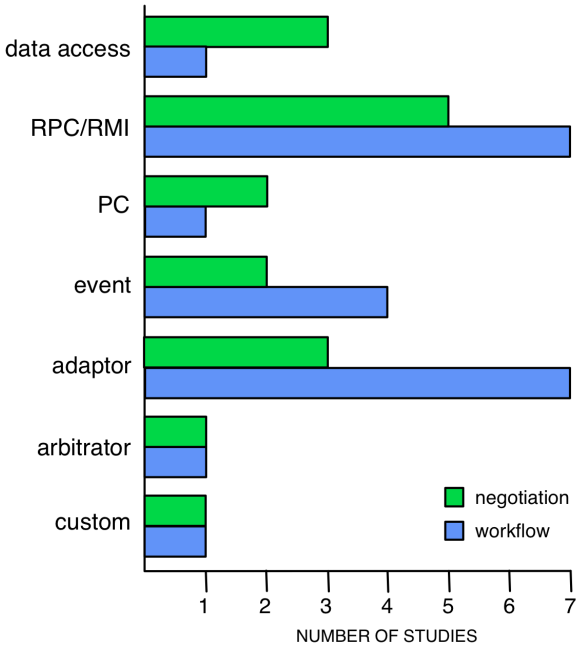

Fig. 24. Negotiation and workflow connectors

Architectural styles

The evaluated architectures are based on a combination of distributed objects, implicit invocation, and layered architectural styles. Most architectures exhibit client-server configurations, but in a couple of instances—specifically eMediator and CrossFlow—the chosen configuration is peer-to-peer. In general, client-server configurations are induced by the presence of business intermediaries. For example, buyer and seller clients register their needs and offerings in the MAGNET *market*.

Not all studies explicitly endorse an overarching architectural style. Where that was the case, we were able to infer a style based on the employed connector types, although in some cases we were not able to distinguish among event-based XML-based messages and remote procedure calls embedded in XML documents (i.e. XML/RPC).

Implementation technologies can also induce a given style. For example building an application with Java induces an object-oriented architecture.

In most cases, we found that a combination of styles were — intentionally or unintentionally — used, perceivable at different levels of abstraction, or discernible in different aspects or places in the application. For example, in E-ADOME agents interact through messages, but they make explicit procedure calls to workflow business objects.

Architectural styles play a secondary role or none within the evaluated studies, being there more focus on the core architectural components and on the implementation technologies.

Only in a few instances, specially in ERA, we found a deeper reflection in architectural concerns, the reasoning behind choosing one style or the other, the use of design patterns, and a clear understanding of the properties induced by those architectural design decisions.

The difficulty of evolving an architecture without the guidance of an architectural style or a consistent set of styles—as well as a clear understanding of the properties those style yield— can lead to architectural erosion [106] and a less cohesive, scalable, understandable, and maintainable system.

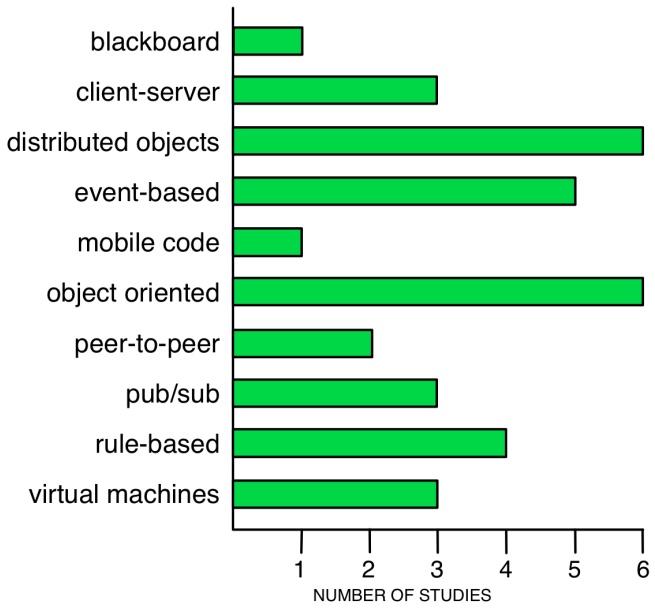

Fig. 25. Architectural styles

Distribution and decentralization

Overall, the evaluated architectures are highly distributed and decentralized given the nature of the domain. The existence of a negotiation process and a contract is evidence of interaction among autonomous parties, thus these technologies are inherently decentralized.

Since negotiation often involves competitive and conflicting interests, distribution of negotiation components is often a requirement since it is unlikely that parties trust another one to host the negotiation process. In addition, there are private and strategic negotiation information that participants do not want to reveal to other parties. Therefore, software agents and web clients negotiate distributedly.

A discussed alternative is to have a third party managing the negotiation infrastructure, enabling non-distributed mobile agent negotiation and thus reducing latency and bandwidth use. However, intermediaries do not prevent the existence of distributed negotiating agents.

There are also hybrid approaches—for example MAGNET—where parties maintain proxies at the negotiation site which receive and send notifications to corresponding distributed agents.

Inter-organizational workflows are inherently decentralized, and the activities of participating organizations are guided by the agreed contract. Each party manages its own workflow through WfMS, and are linked with other workflows in a peer-to-peer style through messaging protocols. For example, in CrossFlow, outsourced processes are initiated through a request message from consumer to service provider. In some instances, participants share a restricted view of their local workflow as we observed in E-ADOME.

Challenges of distributed and decentralized interaction and the absence of a central coordinator are planning and scheduling, since time predictions largely depend on schedules reported by third parties.

Dynamic adaptation and exception handling

Dynamic adaptation allows coping with new requirements and internal or external (intra- or inter-organizational) changes. *This essential and important system property is one of the least supported ones within the evaluated systems and mostly limited to the workflow stage.*

Dynamic adaptation of negotiation strategies is nonexistent in our sample studies and our intuition is that is is also scarcely supported in specialized negotiation systems. ContractBot—specifically AuctionBot—and other negotiation systems support dynamic adaptation of negotiating parties' location through user web interfaces, namely a human moving from one computer to another. However these are manual processes that do not enable negotiation automation. Only eMediator supports dynamic adaptation of parties' location through agent mobility [95].

We identified opportunities where the proposed architectures could be extended to support negotiation adaptation. eMediator's mobile agents could dynamically change their location during negotiation to be closer to their negotiation counterparts. Cheung et. al's negotiation model based on precedence and dependence of variables can be extended to allow on-the-fly modification of the negotiation plan (therefore of the negotiation strategy) according to new information or based on the agreed values of precedent variables. ERA provides a *negotiator* component with access to negotiation policies and rules, providing an opportunity to encode within those rules negotiation strategy adaptation.

Only one study—ERA—considers contract modifications while in the workflow stage through contract updates or "subsidiary arrangements". However, no evaluation prototype is provided. Related to adaptation is the ability to track contract evolution and thus the evolution of the relationship of business partners. However, no study reported support for tracking contract evolution.

Adaptation to changes is largely limited to workflow, where dynamic creation of workflows [63], adaptation to new requirements [63], dynamic process outsourcing [39], and exception handling are forms of adaptation. Although process languages such as Little-JIL can be leveraged to model and potentially drive the execution of software adaptations, there is nothing inherent in this language specific to or explicitly supportive of dynamic adaptation.

Exception handling is the most supported type of adaptation through the redistribution of work in case of unit failure [53], by repairing an outsourcing plan [21], allowing alternative execution paths

[39], or allowing run-time addition or modification of exception handlers [18]. The caveat is that dynamic adaptations are contingent upon known and expected exceptions. Unexpected exceptions usually require human intervention.

A different but related concern is resilience where either negotiation or workflow processes can be recovered from partial or complete software failure. Few studies consider resilience as an architectural properties. In ECBNS the activity log can be leveraged to restart from crashes, while E-ADOME supports exception driven workflow recovery.

Database-like transaction support—discussed in ECBNS E-ADOME and supported in CrossFlow—is a form of combined adaptation and exception handling, where process rollback mechanisms allow undoing the effects of certain actions, thus recovering a prior version of the workflow.

It is important to differentiate between operational and technical exceptions. The first one refers to the failure to obtain the expected information or results (e.g. a delay in procurement). The second one refers to a software failure. Identifying and classifying these potential failures allows designing robust systems which can remediate both types of failures.

The challenges of workflow largely remain the ability to cope and predict changes, improving management and execution of business processes, managing process compositions and tasks decompositions, and the inadequate handling of exceptions [53].

Overall, there is a unmet need to support run-time adaptation of operational and technical requirements — e.g. new business participants — across the e-commerce lifecycle—negotiation, contracts, and workflow—which requires careful analysis of adaptation points and state recovery mechanisms.

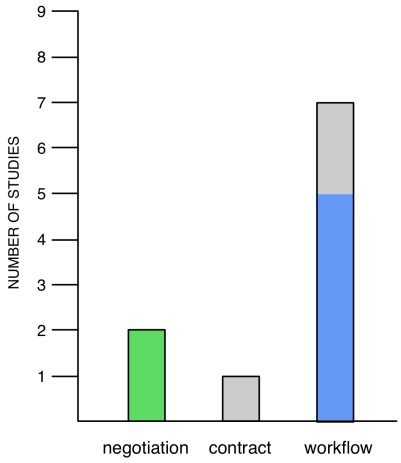

Fig. 26. Dynamic adaptation

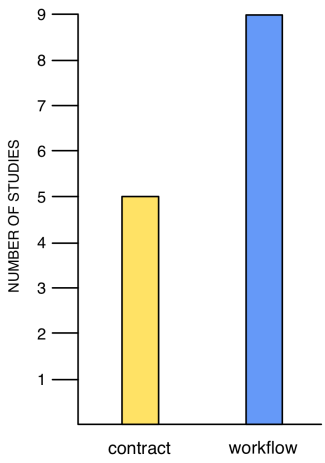

Fig. 27. Exception handling

Automation

Assessing automation support within the evaluated studies was in some cases challenging due to unsatisfactory descriptions of how and when automation is supported and to what extent. Studies are frequently unclear on the context and the specific activities where partial or full automation takes place. Fully automated systems are rare—as well as difficult to build—since recurrently human input and decision making is needed. Also the cost of integration and automation is in general significant. In small enterprises the overhead and cost of building an expert system for e-commerce may not even be worth the effort [40].

Automation is appropriate for mechanical procedures or where components can decide based on a set of rules. It is important to identify when automation or computer support is needed. Also, identifying bottlenecks in business processes provides opportunities for partial or full automation.

Automation is not always applicable or appropriate, for example in situations where human control is more valued. E-commerce applications designers should not assume the degree of control parties expect. Therefore it is important to give the freedom to choose between different levels of automation at different e-commerce stages, such that one party might delegate negotiation to a software agent, while the other prefers human involvement.

In addition, a greater degree of flexibility allows humans to take control over the process at any point, as well as shift from manual intervention to automated action.

While workflow activities exhibit more frequently automation support, automated negotiation is also enabled through autonomous software agents which engage in auctions for offering or purchasing goods, or for task assignment. We have not observed automated support for one-to-one negotiations.

Contract models can also enable automation by the choice of modeling language and by capturing—from a natural language contract—those aspects which are prone to automation.

There is a tradeoff between automation power and complexity of formal contract languages.

While logic based languages allow automation of contract verification and execution, their specification is more time consuming, complex, and hard to understand. XML is a middle-ground approach which allows some degree of automation, but is as well easier to understand and compose. In the literature [89], automation and expressiveness are considered competitors, but a formal language can be as expressive as a semi-formal one such as XML (although more complex).

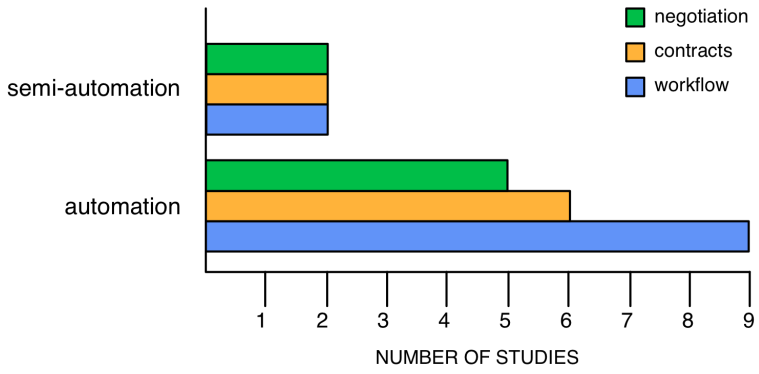

Fig. 28. Automation support

The data format with which business components communicate can enable various levels of automation. For example, Electronic Data Interchange enables heterogenous systems to understand the semantics of the message and respond accordingly with other messages or actions.

Most business processes in the evaluated studies involve some degree of automation, exhibited in a variety of ways and for different purposes. For example, automation of inter-organizational business processes and contract commitments execution, notification of committed activities to other parties, automated workflow configuration, contract management, payments, generation of documents and enactment plans, infrastructure setup, and workflow recovery. Also, automated exception handling is supported when exceptions are known and expected, whereas unexpected exceptions need to be handled by humans, who may be automatically notified.

However, there are still many functionalities within the evaluated systems which have not been automated due to difficulty or because manual input is required. For example, in ER^{EC} the translation from contract clauses to event-condition-action rules is done manually, having no automation tools for this purpose. Input from domain experts is also required. These difficulties raise the question to whether such system is actually usable.

When designing a software system—in this case an commerce application—it is necessary to include descriptions—possibly within sequence or activity diagrams—of those tasks which are automated, and those which require manual input.

Security and accountability

Security has been one of the barriers to adoption of e-commerce systems among individuals and organizations. For successful e-commerce, security needs to be provided by communication channels and trust reinforced between business partners.

An unexpected finding is that neither of the evaluated aspects of security are a major concern during negotiation in the evaluated studies. It might be possible—although we have not conducted further inquiries—that security against unknown business partners are part of the negotiation strategies.

Overall, security, trust, and accountability among parties boils down to the presence of mediating and monitoring intermediaries which control the negotiation or contract management infrastructure. The exception is eMediator, which supports a risk-mitigation strategy for safe exchange among peers.

There is a need to strengthen security not only in client-server e-commerce systems, but in more spontaneous peer-to-peer commerce as well.

Other security provisions are party identification and authority to fulfill a business role in the contract. In addition, workflow views proposed by E-ADOME protect parties' private process information by giving them control to reveal only what is needed to interact and account for the state of their obligations. Access to activity logs also conditions the accountability among parties. Parties may include in the contract their right to request certain information from providers, thus making them accountable for their activities. In relation to security of the communication medium, digital signatures, authorization, proxies, gateways, and message encryption and verification have been proposed. Find in figure 29 a summary of the security mechanisms employed by the number of studies.

Although e-commerce involves critical information and currency, our perception is that security efforts in the evaluated studies are insufficient. While the security mechanisms found in these architectures might be sufficient when used in combination, they are insufficient when used individually as we have seen in many of the evaluated technologies. At least one study does not address security at all. E-commerce systems need to have an integral approach to security to both secure parties from one another and to secure their communication channels taking into account that many security mechanisms are not mutually exclusive, but instead can compensate for each other's weaknesses.

Security needs to be a fundamental and indispensable requirement for e-commerce systems, where trust among business partners, legally valid contracts, privacy of operations, and reliable communication channels are promoted through the underlying technologies.

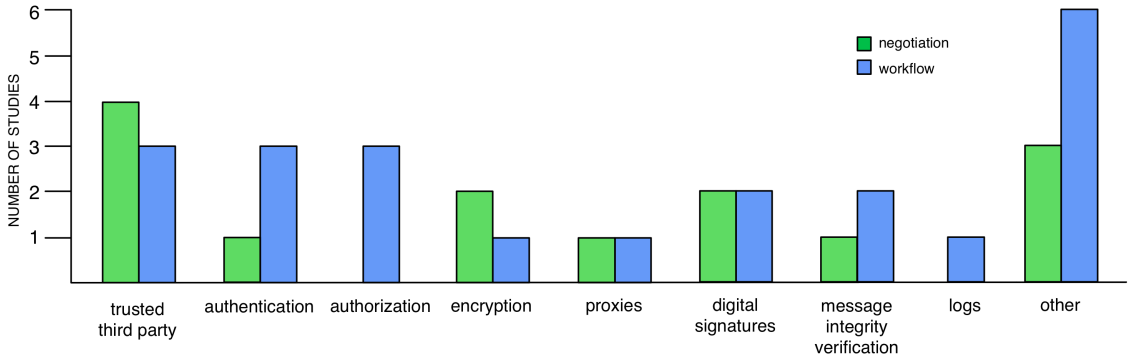

Fig. 29. Security mechanisms in negotiation and workflow

Integration and bi-directionality

These studies demonstrate that these communities are aware of the need to integrate and support the different stages of e-commerce "rather than [having] a collection of independent solutions that may not work in concert" [77].

Our findings show that there is a greater degree of integration between contracts and workflow than contracts and negotiation, and very limited integration between negotiation and workflow.

The logical connection between these domains is that the negotiation process produces a contract which is then executed through workflow technologies. Despite straightforward connection, there are nuances in how these different components interact, as demonstrated in the proposed architectures and domain models.

Negotiation often starts with a contract template whose "blank spaces" are negotiated through auctions or one-to-one negotiation where offers and counteroffers are done through the modification of contract values. The result of this process is a final contract. We were unable to find any meaningful example of bi-directionality between negotiation and contracts, such as the renegotiation and consequent modification of the same contract instance.

Integration between negotiation and workflow is the least supported one, given their indirect relationship, largely mediated by contracts. Workflow configuration or reconfiguration is a byproduct of negotiation results. Having a shared underlying language and technology infrastructure bestows greater integration between negotiation and workflow software components and "bridges the stages by using the output of one stage to formulate the problem of the next" [89]. A different way to link negotiation and workflow concerns is by leveraging workflow technology to drive the negotiation process [17].

We found a few examples of bi-directional relationship between negotiation and workflow. One example is when workflow informs the negotiation process, for example when AEW's agents overbid to inform the contracting agent of their workflow capabilities. The other example is a transitional one, when in the face of exceptions the systems goes back to negotiation to either resolve disputes or find a new workflow participant.

Contracts drive, guide, regulate, distribute assignments, and enable monitoring of business processes. Workflow activities and processes can be explicitly described—fully or partially (views)—in contracts. Workflow specifications and infrastructures can be mapped, configured, or generated—dynamically or manually—from contracts. Greater integration is achieved when an executable contract is the input to a workflow management system.

Bi-directionality between contracts and workflow is more common through contract monitoring and enforcement, given that workflow execution is guided by a contract. Also the service consumer might request changes in the workflow, thus affecting and modifying the corresponding contract.

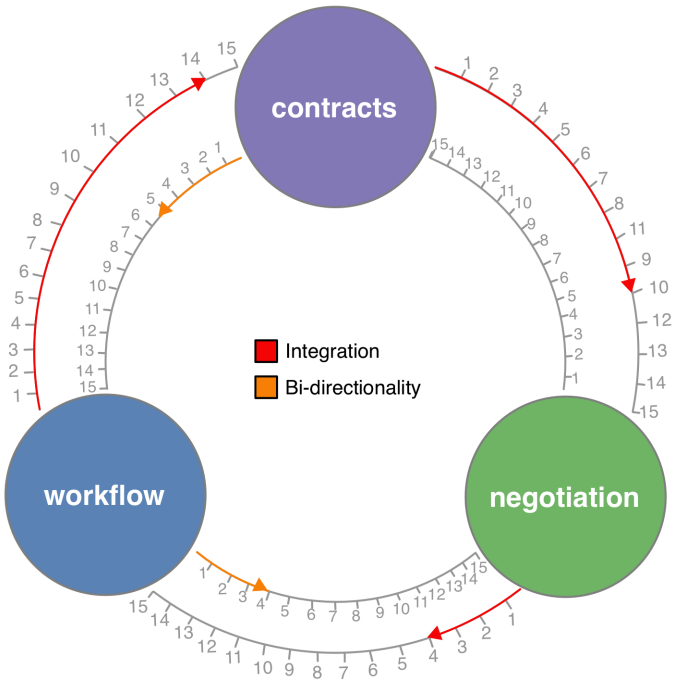

Fig. 30. Integration and bi-directionality

However is not just about the existence of integration, but about the quality, the fluidity, and robustness of such integration. Additionally, the effort and number of steps to get from negotiation to an executed contract can make such integration more or less efficient.

For example, the derivation of business policies and business plan documents in BizTalk B2B is a weaker form of integration than that of MAGNET's explicit task assignment which is the basis for defining execution schedules. An executable contract instance that is input to a WfMS is even a stronger form of integration.

We have observed that while integration is in some cases provided, often the loopholes and layers of indirection to get from one phase to the other can be complex, time consuming, and impractical.

For an example, refer to [63] where going from a contract description to an executable workflow instance involves many layers which include manual steps.

Despite studies reporting integration of the e-commerce contracting lifecycle, *in some cases e-commerce components are listed as a set of requirements, but do not present a concrete description of how they fit together, cooperate, and transition from one phase to the next.*

A clear example is eMediator, which provides an auctioning, a contract optimizer, and an exchange component, but as individual solutions which *could* but they are not explicitly built to work in concert.

Despite that Table 36 shows that AEW, COSMOS, MAGNET, the HP Contract Framework, and eMediator span all three e-commerce domains, *these studies exhibit many shortcomings, such as weak integration among those components, support for narrow business contexts and specific types of interactions—which excludes multi-party negotiation and collaboration—and lack of support for either dynamic adaptation, security, exception handling, or automation, which are important properties for e-commerce applications.*

10 Directions for future work

The insights gained from the structured evaluation of e-commerce technology with respect to the integration between negotiation, contracts, and workflow, as well as of their functional and non-functional properties, open up opportunities for new avenues for inquiry and future research.

With respect to the evaluation framework

There are some ways in which the proposed evaluation framework could be extended and improved for increased comprehensiveness and precision. Our framework can be extended to include the information phase—where consumers search for providers and request for quotes—therefore encompassing the whole e-commerce contracting lifecycle (figure 1). This would provide further insight on the advantages and disadvantages of e-commerce systems with respect to supporting all the stages of e-commerce. In addition, inquiring on the search and information phase can provide insight and comparison on service discovery mechanisms.

In addition, our framework can be extended to establish metrics or typologies to measure and classify different levels or the quality of integration between negotiation, workflow, and contracts. For example, the strength of the relationship between these components could be placed in a defined scale in order to have a conceptual measurement of the integration of the evaluated architectures and systems. This measurement would be a more accurate indicative of the extent to which negotiation, contract, and workflow components interact. A similar scale could be created for automation, security, and so on. The value of these measurements are to obtain a more realistic perspective of how dynamic, secure, and cohesive a software system is.

With respect to the integration of negotiation, contracts, and workflow

Our findings regarding the relation and integration of e-commerce negotiation, contracts, and workflow, as well as their support for distribution, decentralization, dynamic adaptation, automation, security, and accountability, suggest many areas of improvement.

The main avenue for future research is the pragmatic integration of these components of e-commerce. We are specially interested in assessing whether the desired degree of integration can be achieved through the uniformity of an overarching architectural style. An architectural style involves the set of principles which constrain the course of a system's design, inducing beneficial properties in the outcome system. Although a few known architectural styles are discernible from the provided descriptions, only a few studies explicitly ground proposed architectures on specific architectural styles, and in most cases without being framed as a fundamental design decision. Our hypothesis is that by building cohesively and uniformly—and explicitly compared to the evaluated studies—on the principles of an appropriate architectural style, the desired system properties described in our framework can be obtained.

Further work includes exploring the appropriateness of known architectural styles to achieve the desired integration among e-commerce components. Given the nature of e-commerce applications, the various stakeholders involved, and the dynamicity of business relations, an architectural style needs to induce loose coupling and distribution to support the participation of decentralized parties, adaptation given the changing market requirements, security given the sensitive data handled, as well as promote automation to optimize business processes.

In specific, we will be experimenting with the COAST architectural style [36][1] and assessing its suitability for the e-commerce domain, in specific for the integration problem at hand. COAST's principles regarding topology, communication, and security recommends them for building e-commerce applications. COAST's underpinning is computation exchange—as opposed to mere content exchange—among a hierarchy of autonomous peers or *actors* [2]—computational elements deliberately enabled to execute computations—existent within *islands*—unique IP address/port pairs. Computations are addressed by

[1] A detailed description of the COAST architectural style is outside the scope of this survey. Reference to the paper is provided for further architectural and technical details.

Capability URLs (CURLs), which name execution environments and contain authority-to-execute semantics, granting differential access privileges to an actor's binding environment. COAST's authorization and capability-based security model provides controlled access to execution environments, as well as provides support for message unforgeability, self-certification, encryption, and digital signatures.

Our intention is to model organizations as a set of COAST islands and their business assets as a set of collaborating actors which adopt negotiator or business processing roles.

In addition to the style's guiding principles, COAST provides an infrastructure—implemented to support secure mobile code, concurrent computations, and actor *spawning*—which supports building applications adherent to COAST principles. This infrastructure relieves the application designer from dealing with the secure communication mechanisms pertaining to computation exchange and rather focus on the application logic. Our intuition is that these principles and accompanying infrastructure on which to build negotiation, contract, and workflow components will facilitate both their conceptual and technical integration.

In addition, we can leverage COAST's model of computation to explore how to define commercial contracts as computations which are evaluated and exchanged by a set of negotiating and processing actors. We believe a computation is an appropriate way to describe a contract, first because a contract describes a commercial process which can be translated to a formally described algorithm. Second, contracts as computations will be able to represent the state of the relationship between business partners, thus replacing passive XML documents which are limited to describe prescriptive permissions and obligations. A comparison between both types of contract descriptions can be based on expressiveness, dynamicity, understandability, complexity, and ease to compose and use. Third, computations can be transmitted and directly executed without having to go through many levels of translations and conversions. There is a vast literature on contract formalisms on which we can base our work to extend the COAST infrastructure to support the specification of contracts.

Following experimentation may focus on building workflow actors. Application and service customization is highly fostered by COAST's CURLs, which endorse customizable services through the deliberate restriction or access to the capabilities of an execution environment. This capability model suggests appropriate to build business workflow components, enabling service provision with various levels of data access, computability, and fungible resources.

Our intuition is that under this computation model, negotiation actors are no different than workflow actors. The underlying architectural structure—islands, clans, actors, and binding environments—and communication mechanisms—asynchronous messages—are the same whether actors perform negotiation, accounting, or production roles.

In a sample auction scenario a consumer actor can send an expression to be evaluated in various provider actors inviting them to bid for supplying some service. If providers are interested in providing such service they can respond with a *service proposal* computation to be evaluated at the consumer actor and compared to other proposals. The consumer can notify the providers whether a higher bid currently exists. Alternatively, providers can ship long-running computations to the consumer's host which notifies their owners when a higher bid has been submitted. Also, bidding computations can move closer to the consumer host and dynamically adapted in terms of their bidding strategy contingent on changes in the business context and on their organizations' workflow capabilities.

A flexible e-commerce infrastructure needs to allow the configurability of different negotiation interactions or cardinalities, thus experiments on simulating one-to-one, one-to-many, and many-to-many negotiations are part of the same research agenda.

It is worth mentioning that actors and popular negotiation agents are not mutually exclusive, since actors are lower level, autonomous entities which receive or send messages, execute some activity, or create new actors, while agents are application level, goal oriented, continuously running mechanisms which act on behalf of individuals and organizations.

A shortcoming of the evaluated studies is that none of their proposed architectures fully exploits multiple concurrent parties engaged in negotiation of a single contract instance. Further inquires need to explore multi-party scenarios, alongside languages and formalisms to describe and negotiate such agreements. The work by Xu [116] and Van der Aalst [111] can provide insights on how to go about

modeling multi-party contracts for their further integration with negotiation and workflow phases. Also, given the required decision-making autonomy of negotiation actors, our intuition is that a rule-based approach to negotiation—such as the one in [32]—where offer and counteroffer evaluation is based on a set of business rules and constraints is worth exploring.

In addition to the aforementioned research paths, we can explore system scalability, exception handling, and dynamic adaptation of both negotiation and workflow components by way of COAST's actor autonomy, message passing communication, actor spawning, and binding environment *sculpting*—namely deriving binding environments from existing ones. COAST actors can dynamically gain or modify their capabilities by transitively obtaining new binding environments in a message.

While surveying the literature, we perceived a general acceptance of a waterfall-like trend to the stages of e-commerce, namely a model which progresses unidirectionally from negotiation, to contract establishment, and then to contract enactment. The dynamic nature of e-commerce and the way in which these components interact require breaking away from such model and embrace one where all components of e-commerce can be clients and servers of each other, thus exhibiting bi-directional interaction. Therefore, scenarios on the bi-directional relationship between negotiation, contracts, and workflow need to be explored, compared to real world commercial relations, and tested for their feasibility. Examples of such scenarios include going back to previous stages or gathering information from any component belonging to any phase. For example, workflow actors can inform negotiation actors of the current schedule and workload so that negotiations can be planned and carried out accordingly. Given that COAST's actors are autonomous, service providing peers, our assumption is that it is straightforward for a negotiation actor, for example, to request operational capability information to the workflow coordination actor. Another example is when a provider cannot fulfill the terms of the contract, business partners can go back to negotiate the modification of augmentation of the contract.

Another research avenue is exploring whether COAST's authorization and security model is sufficient to protect communication among business partners. Also, parties need to trust each other for successful business endeavors. Suryanarayana et. al [104] present a trust-centric architectural style, study which could provide us further insight on how to promote and manage trust among decentralized business peers.

With respect to implementation concerns, we cannot expect that all business components will be built on *Motile*—COAST's domain-specific, purely functional mobile code language. Moreover the cost of migrating business and legacy components is often too high. We can not disregard the inevitability of system heterogeneity. However to achieve uniformity of the baseline infrastructure, existing business assets and legacy applications can be accessed through wrappers and language-to-Motile compilers. To take advantage of COAST's computation exchange model, other programming languages need to support closures as first class values. Also, the evaluation of a computation within a COAST actor may involve a request for service to an heterogeneous component through the aforementioned wrappers.

In summary, our intention is to assess the appropriateness of the COAST architectural style and whether the desired level of integration can be achieved by building examples of various business scenarios encompassing negotiation, contract, and business workflow components. Our hypothesis is that computations and computational exchange are the integrating commonality and the unifying mechanism for negotiation, contracts, and workflow architectures. In addition, we want to assess whether the beneficial qualities we expect to gain from applying COAST to e-commerce system design are obtained. The overarching goal is enabling communication and collaboration across organizational and geographical boundaries, as well as promoting spontaneous business relationships.

11 Conclusions

E-commerce has made—and continues to make—an immeasurable impact on the commercial and financial industry, opening global markets and changing radically how individuals search, compare, and acquire goods, and how businesses manage their supply chain, procurement, and advertising. E-commerce involves information exchange, collaboration among individuals and organizations, and integration of inter-organizational processes. It is a multifaceted and multidisciplinary domain where technology, finances, marketing, law, economics, sociology, and psychology converge. Needless to say, research in optimizing e-commerce technology is an effort of significant relevance.

We have surveyed the e-commerce literature with the goal of assessing the extent to which integration between fundamental phases or components of e-commerce—negotiation, contracts, and workflow—is supported.

By integration we refer to the high level co-dependent relationship between negotiation, contracts, and workflow and how systems support transitioning from one phase to the next. This high level meaning of integration has specific architectural and implementation level consequences, where negotiation, contracting, and workflow software components interoperate. Integration within e-commerce has so far been used to refer to workflow technologies—including business communication technology such as EDI—and excluding both negotiation and contracts, which are essential components of e-commerce and often responsible for how workflow came into being in the first place. One of our goals is attempt to broaden this contextual meaning of integration, encompassing all the phases of e-commerce.

While surveying the literature, we found that despite the co-dependent relationship between fundamental components of e-commerce—negotiation, contracts, and business workflow—research and development is greatly done in isolation, with minimal overlap of research groups and institutions. Moreover, the evaluated studies which address fully or partially the e-commerce contracting lifecycle are in some cases presented as a set of requirements or as a list of components, but do not address how these components interoperate and work together.

Other studies are successful in describing the steps to progress from one e-commerce stage to the other through a set of cooperating components. However, such integration was often found to be weak or significant effort and many layers of indirection, data transformations, and recurrent manual processes are required to bridge these phases, which negatively impacts performance and cost. This raises the question if these approaches are actually usable and convenient. In addition, in many cases these studies do not provide a prototype to demonstrate the feasibility of their approach.

Studies which bridge two or all of these phases have in general stronger support for workflow than for negotiation. In addition, contracts are the glue that brings together these two phases so there is no study that addresses both negotiation and workflow without addressing contracts.

Our main finding is that despite the existence of successful e-commerce applications, full exploitation of co-dependent business components—negotiation, contracts, and workflow—is elusive. Their smooth integration, automation, and dynamic co-adaptation is a goal, not a current reality.

Also, we found that there are different ways—not necessarily expected and predictable—in which negotiation, contracts, and workflow concerns are configured and used within software architectures. Specific applications have different requirements—often not known in advance—in terms of negotiation cardinality, privacy, security, automation, adaptability, distribution, domain specialization and so on, thus it is necessary to design flexible and securely open systems that enable different kinds of interactions among parties and which can adapt to different kind of requirements. One way to go about doing this is through participatory design, capturing organizations' and individuals' goals and attempting to mediate them through e-commerce technology.

Integration is in general a hard problem whose goal is always difficult to achieve, more so in fields—such as e-commerce—where there are many different stakeholders—from customers, to managers, to producers, to team coordinators, to engineers—abounding information, numerous heterogeneous systems, privacy concerns, and many conflicting interests. In addition, negotiation, contracting, and business workflow are individually complex subfields, let alone their combined interaction. Integration in

this context is hard to achieve with the current state-of-the-art approaches where "components require high set-up efforts if not integrated as a chain of interworking services" [40]. While an important software design principle is that it is not humans who need to adapt to technology, but instead systems need to be designed for human needs, this practice is more challenging in this field given the diversity of individuals and organizations which are used to different ways of conducting business.

Additionally, we assessed important architectural properties—such as distribution, decentralization, dynamic adaptation, security, and automation—within each e-commerce phase or component to identify shortcomings or insufficient support with respect to these important concerns.

Our findings also show insufficient support—nonexistent in some cases—for dynamic adaptation, automation, security, accountability, and bidirectional relationship among e-commerce components in the evaluated systems.

These properties are fundamental within e-commerce given the dynamic nature of commercial relations, the volatility of requirements, the private nature of the exchanged information, and the trust that the technological infrastructure needs to provide to achieve industry adoption.

It is irrefutable that there is an unmet need to support the whole e-commerce contracting lifecycle through the seamless, co-dependent, and bi-directional integration of negotiation, contracting, and business workflow, providing extension mechanisms to support domain specific standards and semantics within contracting and execution activities.

The benefits of this seamless integration in e-commerce systems are satisfying different communication needs among parties, overcoming coordination delays and bureaucratic obstacles, optimizing and automating processes where needed, enabling the dynamic adaptation of systems based on co-dependent e-commerce phases and activities, providing scalability through distribution and decentralization, securely providing activity accountability and communication, and fostering trust among business partners.

References

1. Lawyers, legal forms, law books & software, free legal information - nolo.com. http://www.nolo.com/.
2. Agha, G. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.
3. Albrecht, C. C., Dean, D. L., and Hansen, J. V. Marketplace and technology standards for B2B e-commerce: progress, challenges, and the state of the art. *Information & Management 42*, 6 (2005), 865–875.
4. Allen, T., and Widdison, R. Can computers make contracts. *Harv. JL & Tech. 9* (1996), 25.
5. Angelov, S., and Grefen, P. The 4W framework for B2B e-contracting. *International journal of networking and virtual organisations 2*, 1 (2003), 78–97.
6. Angelov, S., and Grefen, P. An e-contracting reference architecture. *Journal of Systems and Software 81*, 11 (Nov. 2008), 1816–1844.
7. Basu, A., and Kumar, A. Research commentary: Workflow management issues in e-business. *Information Systems Research 13*, 1 (2002), 114.
8. Beugnard, A., Jezequel, J. M., Plouzeau, N., and Watkins, D. Making components contract aware. *Computer 32*, 7 (1999), 38–45.
9. Bichler, M., Kersten, G., and Strecker, S. Towards a structured design of electronic negotiations. *Group Decision and Negotiation 12*, 4 (2003), 311–335.
10. Blake, M. B2B electronic commerce: Where do agents fit in? In *Proceedings of the AAAI-2002 Workshop on Agent Technologies for B2B E-Commerce, Edmonton, Alberta, Canada* (2002).
11. Boulmakoul, A., and Sall, M. Integrated contract management. In *Proceedings of the 9th Workshop of the HP OpenView University Association* (2002).
12. Buttner, R. A classification structure for automated negotiations. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops* (Dec. 2006), IEEE, pp. 523–530.
13. Carmo, J., and Jones, A. Deontic logic and contrary-to-duties. *Handbook of philosophical logic 8* (2002), 265–343.
14. Cass, A. G., Lee, H., Lerner, B. S., and Osterweil, L. J. Formally defining coordination processes to support contract negotiation. Tech. rep., University of Massachusetts, Amherst, MA, USA, 1999.
15. Cass, A. G., Lerner, B. S., Stanley M. Sutton, J., McCall, E. K., Wise, A., and Osterweil, L. J. Little-JIL/Juliette: a process definition language and interpreter. In *Proceedings of the 22nd international conference on Software engineering* (Limerick, Ireland, 2000), ACM, pp. 754–757.
16. Chavez, A., and Maes, P. Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* (1996), vol. 31, p. 40.
17. Cheung, S., Hung, P., and Chiu, D. A meta-model for e-Contract template variable dependencies facilitating e-Negotiation. *Conceptual ModelingER 2002* (2003), 50–64.
18. Chiu, D., Karlapalem, K., Li, Q., and Kafeza, E. Workflow view based e-contracts in a cross-organizational e-services environment. *Distributed and Parallel Databases 12*, 2 (2002), 193–216.
19. Chiu, D. K., Cheung, S., Till, S., Karlapalem, K., Li, Q., and Kafeza, E. Workflow view driven Cross-Organizational interoperability in a web service environment. *Information Technology and Management 5*, 3/4 (July 2004), 221–250.
20. Chiu, D. K., Li, Q., and Karlapalem, K. A meta modeling approach to workflow management systems supporting exception handling. *Information Systems 24*, 2 (1999), 159–184.
21. Collins, J., Tsvetovat, M., Mobasher, B., and Gini, M. MAGNET: a multi-agent contracting system for plan execution. In *Proceedings of SIGMAN* (1998), pp. 63–68.
22. Damanpour, F., and Damanpour, J. A. E-business e-commerce evolution: perspective and strategy. *Managerial Finance 27*, 7 (Jan. 2001), 16–33.
23. Daskalopulu, A. Modelling legal contracts as processes. In *11th International Workshop on Database and Expert Systems Applications, 2000. Proceedings* (2000), IEEE, pp. 1074–1079.
24. Daskalopulu, A., and Maibaum, T. Towards electronic contract performance. In *Proceedings of the 12th International Workshop on Database and Expert Systems Applications* (2001), IEEE, pp. 771–777.
25. Daskalopulu, A., and Sergot, M. The representation of legal contracts. *AI & Society 11*, 1 (1997), 6–17.
26. Dumas, M., and ter Hofstede, A. UML activity diagrams as a workflow specification language. *UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools* (2001), 76–90.
27. Eiselen, S. Electronic commerce and the UN convention on contracts for the international sale of goods (CISG) 1980. *EDI Law Review 6* (1999), 21–46.
28. Ellis, C. Team automata for groupware systems. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge* (1997), pp. 415–424.
29. Fakas, G., and Karakostas, B. A peer to peer (P2P) architecture for dynamic workflow management. *Information and Software Technology 46*, 6 (2004), 423–431.

30. Farrell, A. D., Sergot, M. J., Salle, M., and Bartolini, C. Using the event calculus for tracking the normative state of contracts. *International Journal of Cooperative Information Systems 14*, 2-3 (2005), 99–130.
31. Frikken, K., and Atallah, M. Achieving fairness in private contract negotiation. *Financial Cryptography and Data Security* (2005), 270–284.
32. Ganzha, M., and Paprzycki, M. Implementing rule-based automated price negotiation in an agent system. *Journal of Universal Computer Science 13*, 2 (2007), 244–266.
33. Georgakopoulos, D., Hornick, M., and Sheth, A. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases 3*, 2 (1995), 119–153.
34. Gisler, M., Stanoevska-Slabeva, K., and Greunz, M. Legal aspects of electronic contracts. *Ludwig, H.; Hoffner, Y.; Bussler, C* (2000), 56.
35. Glushko, R., Tenenbaum, J., and Meltzer, B. An XML framework for agent-based e-commerce. *Communications of the ACM 42*, 3 (1999), 106–ff.
36. Gorlick, M., Strasser, K., Baquero, A., and Taylor, R. N. CREST: principled foundations for decentralized systems. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion* (2011), pp. 193–194.
37. Governatori, G. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems 14*, 2-3 (2005), 181–216.
38. Governatori, G., and Milosevic, Z. A formal analysis of a business contract language. *International Journal of Cooperative Information Systems 15*, 4 (2006), 659–685.
39. Grefen, P., Aberer, K., Hoffner, Y., and Ludwig, H. CrossFlow: Cross-Organizational workflow management in dynamic virtual enterprises.
40. Griffel, F., Boger, M., Weinreich, H., Lamersdorf, W., and Merz, M. Electronic contracting with COSMOS-how to establish, negotiate and execute electronic contracts on the internet. In *Proceedings of the Second International Enterprise Distributed Object Computing Workshop. EDOC'98.* (1998), pp. 46–55.
41. Grosof, B., Labrou, Y., and Chan, H. A declarative approach to business rules in contracts: courteous logic programs in XML. In *Proceedings of the 1st ACM conference on Electronic commerce* (1999), pp. 68–77.
42. Grosof, B. N., and Poon, T. C. SweetDeal: representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In *Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), WWW '03, ACM, pp. 340–349.
43. Gulledge, T. What is integration? *Industrial Management & Data Systems 106*, 1 (2006), 5–20.
44. Guttman, R., and Maes, P. Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. *Cooperative Information Agents II Learning, Mobility and Electronic Commerce for Information Discovery on the Internet* (1998), 135–147.
45. Guttman, R., Moukas, A., and Maes, P. Agent-mediated electronic commerce: a survey. *The Knowledge Engineering Review 13*, 02 (1998), 147–159.
46. Guttman, R. H., and Maes, P. Agent-Mediated integrative negotiation for retail electronic commerce. In *Agent Mediated Electronic Commerce*, vol. 1571. Springer, Berlin, Heidelberg, 1999, pp. 70–90.
47. He, M., Jennings, N. R., and Leung, H. On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering 15*, 4 (2003), 985–1003.
48. Herring, C., and Milosevic, Z. Implementing B2B contracts using BizTalk. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on* (2001), pp. 10–pp.
49. Hillman, R., and Rachlinski, J. Standard-form contracting in the electronic age. *NYUL Rev. 77* (2002), 429.
50. Hsu, M., and Kleissner, C. ObjectFlow: towards a process management infrastructure. *Distributed and Parallel Databases 4*, 2 (1996), 169–194.
51. Hvitved, T. A survey of formal languages for contracts.
52. Jablonski, S. MOBILE: a modular workflow model and architecture. In *Proc. of Int'l Working Conference on Dynamic Modelling and Information Systems, Nordwijkerhout* (1994).
53. Judge, D., Odgers, B., Shepherdson, J., and Cui, Z. Agent-enhanced workflow. *BT Technology Journal 16*, 3 (1998), 79–85.
54. Kammer, P. J., Bolcer, G. A., Taylor, R. N., Hitomi, A. S., and Bergman, M. Techniques for supporting dynamic and adaptive workflow. *Computer Supported Cooperative Work (CSCW) 9*, 3 (2000), 269–292.
55. Keller, A., and Ludwig, H. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management 11*, 1 (2003), 57–81.
56. Kersten, G., and Lai, H. Negotiation support and e-negotiation systems: an overview. *Group Decision and Negotiation 16*, 6 (2007), 553–586.
57. Kersten, G., and Lo, G. Aspire: an integrated negotiation support system and software agents for e-business negotiation. *International Journal of Internet and Enterprise Management 1*, 3 (2003), 293–315.

58. Khare, R., and Taylor, R. N. Extending the representational state transfer (rest) architectural style for decentralized systems. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on* (2004), pp. 428–437.
59. Kidd, D., and Daughtrey Jr, W. Adapting contract law to accommodate electronic contracts: Overview and suggestions. *Rutgers Computer & Tech. LJ 26* (1999), 215.
60. Ko, R., Lee, S., and Lee, E. Business process management (BPM) standards: a survey. *Business Process Management Journal 15*, 5 (2009), 744–791.
61. Kowalczyk, R., Ulieru, M., and Unland, R. Integrating mobile and intelligent agents in advanced e-commerce: a survey. *Agent Technologies, Infrastructures, Tools, and Applications for E-Services* (2003), 295–313.
62. Krishna, P., and Karlapalem, K. Electronic contracts. *Internet Computing, IEEE 12*, 4 (2008), 60–68.
63. Krishna, P. R., Karlapalem, K., and Chiu, D. An ER-EC framework for e-contract modeling, enactment and monitoring. *Data & Knowledge Engineering 51*, 1 (2004), 31–58.
64. Krishnakumar, N., and Sheth, A. Managing heterogeneous multi-system tasks to support enterprise-wide operations. *Distributed and Parallel Databases 3*, 2 (Apr. 1995), 155–186.
65. Kumar, M., and Feldman, S. Internet auctions. In *Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce* (1998), vol. 3, p. 55.
66. Kyas, M., Prisacariu, C., and Schneider, G. Run-Time monitoring of electronic contracts. In *Automated Technology for Verification and Analysis*, vol. 5311. Springer, Berlin, Heidelberg, 2008, pp. 397–407.
67. Leymann, F., and Roller, D. Business process management with flowmark. In *Compcon Spring'94, Digest of Papers.* (1994), pp. 230–234.
68. Lomuscio, A., Wooldridge, M., and Jennings, N. A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation 12*, 1 (2003), 31–56.
69. Maes, P., Guttman, R., and Moukas, A. Agents that buy and sell. *Communications of the ACM 42*, 3 (1999), 81–ff.
70. Malone, T. The future of work. In *Designing Ubiquitous Information Environments: Socio-Technical Issues and Challenges*, C. Srensen, Y. Yoo, K. Lyytinen, and J. DeGross, Eds., vol. 185 of *IFIP International Federation for Information Processing*. Springer Boston, 2005, pp. 17–20.
71. Marchetti, A., Tesconi, M., and Minutoli, S. Xflow: An xml-based document-centric workflow. *Web Information Systems Engineering. WISE 2005* (2005), 290–303.
72. Marjanovic, O., and Milosevic, Z. Towards formal modeling of e-Contracts. In *Enterprise Distributed Object Computing Conference, IEEE International* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 59–68.
73. McCarthy, D., and Sarin, S. Workflow and transactions in inconcert. *IEEE Data Engineering 16*, 2 (1993), 53–56.
74. Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. The action workflow approach to workflow management technology. *The Information Society 9*, 4 (1993), 391–404.
75. Mehta, N., Medvidovic, N., and Phadke, S. Towards a taxonomy of software connectors. In *Proceedings of the 22nd international conference on Software engineering* (2000), pp. 178–187.
76. Meyer, J. J., and Wieringa, R. J. Applications of deontic logic in computer science: A concise overview.
77. Milosevic, Z., Berry, A., Bond, A., and Raymond, K. Supporting business contracts in open distributed systems. In *Services in Distributed and Networked Environments, 1995., Second International Workshop on* (1995), pp. 60–67.
78. Milosevic, Z., Josang, A., Dimitrakos, T., and Patton, M. Discretionary enforcement of electronic contracts. In *Proceedings of the Sixth International Enterprise Distributed Object Computing Conference, 2002. EDOC'02* (2002), pp. 39–50.
79. Molina-Jimenez, C., Shrivastava, S., Solaiman, E., and Warne, J. Run-time monitoring and enforcement of electronic contracts. *Electronic Commerce Research and Applications 3*, 2 (2004), 108–125.
80. Muller, R., Greiner, U., and Rahm, E. AW: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering 51*, 2 (2004), 223–256.
81. Muth, P., Wodtke, D., Weissenfels, J., Dittrich, A., and Weikum, G. From centralized workflow specification to distributed workflow execution. *Journal of Intelligent Information Systems 10*, 2 (1998), 159–184.
82. Oprea, M. An adaptive negotiation model for agent-based electronic commerce. *Studies in Informatics and Control 11*, 3 (2002), 271–279.
83. Oreizy, P., Gorlick, M., Taylor, R., Heimhigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D., and Wolf, A. An architecture-based approach to self-adaptive software. *Intelligent Systems and their Applications, IEEE 14*, 3 (June 1999), 54 –62.

84. Papazoglou, M. P., Jeusfeld, M. A., Weigand, H., and Jarke, M. Distributed, interoperable workflow support for electronic commerce. In *Trends in Distributed Systems for Electronic Commerce*, vol. 1402. Springer-Verlag, Berlin/Heidelberg, 1998, pp. 192–204.
85. Parsons, S., and Wooldridge, M. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems 5*, 3 (2002), 243–254.
86. Prisacariu, C., and Schneider, G. A formal language for electronic contracts. In *Formal Methods for Open Object-Based Distributed Systems*, vol. 4468. Springer, Berlin, Heidelberg, 2007, pp. 174–189.
87. Prisacariu, C., and Schneider, G. CL: an action-based logic for reasoning about contracts. In *Logic, Language, Information and Computation: 16th International Workshop, WoLLIC 2009, Proceedings* (Tokyo, Japan, June 2009), vol. 5514, p. 335.
88. Rahwan, I., Ramchurn, S., Jennings, N., Mcburney, P., Parsons, S., and Sonenberg, L. Argumentation-based negotiation. *The Knowledge Engineering Review 18*, 4 (2003), 343–375.
89. Reeves, D., Wellman, M., and Grosof, B. Automated negotiation from declarative contract descriptions. *Computational Intelligence 18*, 4 (2002), 482–500.
90. Ren, J., Taylor, R., Dourish, P., and Redmiles, D. F. Towards an architectural treatment of software security: a connector-centric approach. *ACM Sigsoft Software Engineering Notes 30*, 4 (2005), 17.
91. Rohwer, C. D., and Schaber, G. D. *Contracts in a Nutshell*. West Publishing Company, 1997.
92. Sadiq, S., Marjanovic, O., and Orlowska, M. Managing change and time in dynamic workflow processes. *International Journal of Cooperative Information Systems 9*, 1-2 (2000), 93–116.
93. Sandholm, T. Automated negotiation. *Communications of the ACM 42*, 3 (1999), 84–85.
94. Sandholm, T. Agents in electronic commerce: Component technologies for automated negotiation and coalition formation. *Autonomous Agents and Multi-Agent Systems 3*, 1 (2000), 73–96.
95. Sandholm, T. eMediator: a next generation electronic commerce server. *Computational Intelligence 18*, 4 (2002), 656–676.
96. Sarin, D. Workflow and transactions in InConcert. *Bulletin of the Technical Committee on 16*, 2 (1993), 53.
97. Schafer, J., Konstan, J., and Riedl, J. E-commerce recommendation applications. *Data mining and knowledge discovery 5*, 1 (2001), 115–153.
98. Scheer, A., and Nttgens, M. ARIS architecture and reference models for business process management. *Business Process Management* (2000), 301–304.
99. Schulz, K., and Orlowska, M. Facilitating cross-organisational workflows with a workflow view approach. *Data & Knowledge Engineering 51*, 1 (2004), 109–147.
100. Shim, S., Pendyala, V., Sundaram, M., and Gao, J. Business-to-business e-commerce frameworks. *Computer 33*, 10 (2000), 40–47.
101. Skylogiannis, T., Antoniou, G., Bassiliades, N., Governatori, G., and Bikakis, A. DR-NEGOTIATE-A system for automated agent negotiation with defeasible logic-based strategies. *Data & Knowledge Engineering 63*, 2 (2007), 362–380.
102. Smith, R. The contract net protocol: High-level communication and control in a distributed problem solver. *Computers, IEEE Transactions on 100*, 12 (1980), 1104–1113.
103. Suchman, L. Technologies of accountability: of lizards and aeroplanes. *Technology in working order* (1993), 113–126.
104. Suryanarayana, G., Erenkrantz, J. R., Hendrickson, S. A., and Taylor, R. N. PACE: an architectural style for trust management in decentralized applications. In *Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on* (2004), pp. 221–230.
105. Sycara, K. Argumentation: Planning other agents' plans. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (1989), pp. 517–523.
106. Taylor, R. N., Medvidovic, N., and Dashofy, E. M. *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, 2010.
107. Van der Aalst, W. Process-oriented architectures for electronic commerce and interorganizational workflow. *Information Systems 24*, 8 (1999), 639–671.
108. Van Der Aalst, W. Business process management demystified: A tutorial on models, systems and standards for workflow management. *Lectures on Concurrency and Petri Nets* (2004), 21–58.
109. Van der Aalst, W., Ter Hofstede, A., Kiepuszewski, B., and Barros, A. Workflow patterns. *Distributed and parallel databases 14*, 1 (2003), 551.
110. Van der Aalst, W., ter Hofstede, A., and Weske, M. Business process management: A survey. *Business Process Management* (2003), 1019–1019.
111. Van Der Aalst, W. M., Lohmann, N., Massuthe, P., Stahl, C., and Wolf, K. Multiparty contracts: Agreeing and implementing interorganizational processes. *The Computer Journal 53*, 1 (2010), 90–106.
112. Von Wright, G. Deontic logic. *Mind 60*, 237 (1951), 115.
113. Watnick, V. Electronic formation of contracts and the common law mainbox rule, the. *Baylor L. Rev. 56* (2004), 175.

114. Wodtke, D., Wei\ssenfels, J., Weikum, G., and Dittrich, A. The mentor project: Steps towards enterprise-wide workflow management. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on* (1996), pp. 556–565.
115. Wurman, P. R., Wellman, M. P., and Walsh, W. E. The michigan internet AuctionBot: a configurable auction server for human and software agents. In *Proceedings of the second international conference on Autonomous agents* (New York, NY, USA, 1998), AGENTS '98, ACM, pp. 301–308.
116. Xu, L. A multi-party contract model. *ACM SIGecom Exchanges 5*, 1 (2004), 13–23.