Available online at www.sciencedirect.com

ScienceDirect

The Journal of Systems and Software 81 (2008) 1816–1844

The Journal of Systems and Software

www.elsevier.com/locate/jss

An e-contracting reference architecture

Samuil Angelov *, Paul Grefen

Faculty of Technology Management, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands

Received 1 June 2007; received in revised form 27 November 2007; accepted 16 February 2008
Available online 26 February 2008

Abstract

Business-to-business e-contracting aims at automating the contracting process between companies. It improves the efficiency of the contracting process and enables the introduction of new business models that can be supported by companies. For the development of an e-contracting system, an architecture is required that describes the system components and the communication channels between them. This paper presents a reference architecture for the development of e-contracting systems. The architecture is designed on the basis of a requirement analysis of e-contracting systems. Established architectural principles are used in its design. The architecture can serve as a foundation in the analysis and design of concrete architectures of e-contracting systems. Furthermore, it can be used as a standardization model that facilitates system integration and communication of ideas. Its value for both software architects and business professionals makes it an important tool in the analysis and implementation of e-contracting systems.

Keywords: E-contracting; Electronic contract; E-contracting architecture; Reference architecture

1. Introduction

Since the very beginning of human history, people have been exchanging values. Contracts between value-exchanging sides have been adopted to specify the exchanged values and the rights and obligations of the participants. Nowadays, contracts are an indispensable tool in business exchanges: "all economic production and exchange processes are organized through contracts. Contracts are the instruments and the means for the organization of exchange relations" (Wigand et al., 1997).

Business-to-business e-contracting uses information technology for improving the efficiency and effectiveness of contracting processes of companies. One way to implement e-contracting is by simply digitizing existing paper contracts and using fast communication channels for contract establishment (e.g., e-mail). We call this type of e-contracting "shallow e-contracting" (Angelov and Grefen, 2004b). Shallow e-contracting improves the efficiency of the contracting process by reducing the time and costs for communication. However, shallow e-contracting requires significant human involvement and does not change traditional business and organizational models. Shallow e-contracting can be supported by existing and widely accepted information technology (e-mail clients, text editors, etc.). The second way to support e-contracting is by implementing a dedicated e-contracting system that can fully (or to a great extent) automate the e-contract establishment, enactment, and management. We call this type of e-contracting "deep e-contracting" (Angelov and Grefen, 2004b). Deep e-contracting eliminates (or significantly decreases) human participation in contracting processes. The high level of automation of contracting processes allows companies to realize new business and organizational models that lead to improved market competitiveness. Deep e-contracting allows, for example, the support of contracting in the latest possible moment (just-in-time contracting) and of contracting of micro business relationships (micro-contracting). An elaborate discussions on the values introduced by deep e-contracting

* Corresponding author. Tel.: +31 40 247 2617; fax: +31 40 243 2612.
E-mail addresses: s.angelov@tue.nl (S. Angelov), p.w.p.j.grefen@tue.nl (P. Grefen).

0164-1212/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2008.02.023

is presented in Angelov and Grefen (2004b). A number of business domains allow high automation of contracting relationships (e.g. on-line advertising (Angelov and Grefen, 2006)). Businesses from such domains that want to benefit from one or more of the values introduced by deep e-contracting will have to implement an advanced e-contracting system that can automate its contracting related processes.

As in most modern software development projects, the development and implementation of a "deep e-contracting system" must be preceded by the effort- and time-consuming "system analysis" and "system design" phases (Maciaszek, 2001). The existence of a specification of the requirements on a highly-automated e-contracting system and of a reference architecture to serve as a guideline in the design of concrete architectures will significantly facilitate the software development process of deep e-contracting systems. A reference architecture for a deep e-contracting system will bring a number of other benefits as well. Similar to other reference architectures (e.g., Hollingsworth, 1995), it will facilitate modular configuration of e-contracting systems. Different vendors will be able to provide specific modules that can be easily integrated. It will facilitate interoperability of deep e-contracting systems with other information systems. It will make easier the analysis and evaluation of existing e-contracting systems. Last but not least, its existence will provide a standardized view on e-contracting systems which will facilitate communications between the potential stakeholders (business professionals, software developers).

A few efforts for the design of complete or partial e-contracting architectures exist. Many of these efforts propose concrete e-contracting architectures that are suitable for specific business situations and software solutions (e.g., Dan et al., 1998; Hoffner et al., 2001a; Ludwig et al., 2004). Efforts that are independent of concrete technology and business scenarios lack sufficient level of detail and completeness (e.g., Boulmakoul and Salle, 2002; Chiu et al., 2003; Griffel et al., 1998; Milosevic and Bond, 1995).

In this paper, we propose a reference architecture for highly automated e-contracting systems. An e-contracting reference architecture should specify in a detailed way the functionalities that must be delivered by an e-contracting system and should provide the major system design principles. In Kruchten (1995), four views on a system are suggested, i.e., logical, process, implementation, and deployment. The E-contracting Reference Architecture that is presented in this paper (referred from now as ERA) aims at facilitating the design of the logical view of concrete e-contracting systems. The logical view (also called functional (Rozanski and Woods, 2005)) describes the functionalities that the system should provide and is traditionally the starting point for defining an architecture (Rozanski and Woods, 2005). ERA is defined on the basis of a set of required qualities. These qualities were either extracted from existing publications on e-contracting (see Section 2) or were discovered during the initial stages of the design of ERA (see Section 7).

The paper is structured as follows. In Section 2, the identified functional, non-functional, and architecture qualities are presented. The main design principles that are used in the design of ERA are discussed in Section 3. ERA is presented at three levels of abstraction. Descriptions of the first, second, and third levels of ERA are provided in Sections 4–6, respectively. In Section 7, we present an evaluation of ERA and discuss related work. The paper ends with conclusions.

2. Required qualities in a reference architecture for e-contracting

In this section, we discuss the functional and non-functional qualities that must be addressed in ERA. Functional qualities express the functionalities that must be supported by an e-contracting system. We used Angelov and Grefen (2004a) for the definition of the required functionalities. Non-functional qualities are separated into two groups, i.e., system qualities and architecture qualities. The system qualities are qualities that must be addressed in the development of an e-contracting *system*. The architecture qualities are qualities that are important for the design of a "good" e-contracting *reference architecture*. Initially, as an inspiration for the definition of non-functional qualities required in an e-contracting reference architecture, we used the list of non-functional qualities on information systems presented in Bass et al. (2003) and existing publications on e-contracting (Angelov and Grefen, 2002, 2003b, 2004a, 2005; Angelov et al., 2005; Grefen et al., 2003; Griffel et al., 1998; Hoffner et al., 2001b; Merz et al., 1998). Based on our initial list of requirements, we defined a draft version of ERA. During the evaluation of ERA (see Section 7), the list of required non-functional qualities in ERA evolved. In this section, we present the final list of identified qualities. Results from our initial efforts can be found in Angelov (2006, 2007b).

2.1. Required functional qualities

As discussed in Angelov and Grefen (2002), an e-contracting process consists of four phases, i.e., the information, pre-contracting, contracting, and enactment phases. Thus, an e-contracting system must provide support for each of these four phases. In addition, an e-contracting system must provide support for the seamless integration of these phases into a coherent e-contracting process. The following general functional requirements follow from this high-level view on the e-contracting process.

- To support the information phase (finding potential partners), an e-contracting system must provide *matching functionalities*.
- To support the pre-contracting phase (selecting preferred partners for negotiation), an e-contracting system must provide *partner-selection functionalities*.

- To support the contracting phase (negotiating and signing the agreed upon contract), an e-contracting system must provide *negotiation and contract establishment functionalities*.
- To support the enactment phase, an e-contracting system must provide *contract enactment functionalities*.
- To support the integration of the e-contracting phases into a coherent e-contracting process, an e-contracting system must provide *management functionalities*.

In businesses where the set of potential partners is well known and does not change in time, the matchmaking phase does not have to be performed and consequently matchmaking functionalities are not required in e-contracting systems. Furthermore, in certain businesses, partners may even be fixed for long periods of time (e.g., a value-chain with fixed partners). In this case, both the information and pre-contracting phases do not have to be performed and consequently matchmaking and partner selection functionalities are not required in e-contracting systems.

Each e-contracting phase can be decomposed into "abstract" activities. Abstract activities can be decomposed to "concrete" activities. In Table 1, based on Angelov and Grefen (2004a), we provide decomposition of the phases into abstract activities and decomposition of the abstract activities into concrete activities. Table 1 contains the subset of concrete activities that together with the abstract activities are used for refinement of the general functional requirements. The complete set of concrete activities can be found in Angelov and Grefen (2004a).

Table 1
Decomposition of the e-contracting phases

Phases	Abstract activities	Concrete activities
Information	General preparations	Elaborate general provisions, Prepare templates, etc.
	Partner matching	Publish advertisement, Search for advertisements, etc.
Pre-contracting	Partner information	Request information, Send/Receive information, etc.
	Offer	Request offer, Send/Receive offer, etc.
	Partner selection	Evaluate partner, Select partner, etc.
Contracting	Negotiation	Request for contract offer, Reject/Accept contr. offer, etc.
	Signing and storing	Sign, Store internally/externally
Enactment	Value exchange	Request/Send enactment data, Deliver service/reward, etc.
	Monitor and control	Monitor, Control, Notify
	Dispute resolution	Notify for dispute, Request/Send dispute data, etc.
	Evaluation	Evaluate, Store/send evaluation

In addition to the theoretical findings on required functionalities, during the evaluation of initial versions of ERA (see Section 7), we discussed the required functionalities with contract management professionals representing different business domains (Angelov, 2007a). These discussions helped us to confirm and slightly improve the theoretical results on the identified concrete activities. We revisit the results obtained from these discussions and their influence on the architecture in Section 7.

2.2. Required non-functional system qualities

According to Bass et al. (2003), system qualities can be divided into two classes, i.e., system qualities discernable at runtime, and not discernable at runtime. The qualities defined in this section follow the terminology of the information systems domain (Bass et al., 2003). For clarity, we indicate in brackets the corresponding terms in the software engineering domain as defined in International Organization for Standardization (2006), as well as the terms used by non-IT stakeholders (contract managers, contract engineers, lawyers, etc.).

2.2.1. System qualities discernable at runtime

Security (*trust*). An e-contracting system involves the storing and exchange of data with high degree of privacy (Angelov et al., 2005). This requires e-contracting systems to provide a high level of security. In ERA, we address only security functionalities that are characteristic for e-contracting systems. Clearly, message recipients should be able to verify the identity of a message sender in a non-repudiatable way. Communications should be protected from eavesdropping and alterations to messages during their transmission should be detectable. These security issues are usually referred to as "*authentication*", "*non-repudiation*", "*encryption*", and "*integrity*" (Turban et al., 2000). They are commonly agreed to be paramount for e-commerce systems and must be addressed in the design of an e-contracting system as well. Basic security issues like authorization and access control to an e-contracting system are out of the scope of ERA. Existing security frameworks can be used for this purpose (Dhillon, 2007). As indicated by the term "trust", in the context of e-contracting, security can be discussed in a broader, non-IT sense. An e-contracting system must be a trustworthy business solution. Incorporated business intelligence and business data must result in a business behavior desired by the company. Trust can be influenced from outside the system as well, e.g., by trusted third parties (e-notaries, certificate authorities, etc.).

High automation. Companies require contracting systems that provide a high-level of automation of the contract creation, enactment, and management. As already stated in Section 1, we discuss a reference architecture for highly automated e-contracting systems.

Flexibility. Contracting is a highly dynamic process that involves the execution of diverse activities, the participation

of diverse partners, and the exchange of diverse data (Angelov and Grefen, 2004a). Consequently, an e-contracting system must be able to support diverse contracting scenarios with diverse business partners.

A number of other qualities like *performance* (*efficiency*), *portability*, and *reliability* are of importance for an e-contracting system. E-contracting may increase the number of contracts in a company. It may require establishment and execution of contracts in seconds (Angelov and Grefen, 2004b). In such cases, the computational and communication loads on e-contracting systems become considerable (Angelov et al., 2005) and performance of e-contracting systems is of high importance. As an e-contracting system supports the execution of business processes that are vital for a company, the reliability quality is of high importance as well. Similarly, a company would prefer a system that is independent on the underlying hardware and software and thus can be easily ported to new machines and operating systems. These qualities however are out of the scope of ERA, which is aimed at the description of the functionalities of an e-contracting system and the main design principles for the development of an e-contracting system (see Section 1).

2.2.2. System qualities not discernable at runtime

Modifiability (*maintainability*). As e-contracting is a new concept, and currently no advanced e-contracting systems exist, it can be expected that newly developed e-contracting systems will require changes (upgrades, extensions, etc.). Moreover, commercial software regularly undergoes updates and releases of new versions. In addition, the business environment is dynamic and changes in it may lead to the need for changes in the e-contracting system as well. That is why modifications driven by software or business development must be easily applied upon an e-contracting system. Loose coupling of components is of paramount importance to achieve system modifiability.

Integrability. As most modern, complex information systems, e-contracting systems will consist of software modules which will often be developed separately and integrated at a later stage. For this reason, it should be easy to integrate the components of an e-contracting system. On the highly abstract level of reference architectures, integrability of components is achieved by specifying a number of elements that facilitate integration among components (Immonen et al., 2005). These elements might, for example, translate incoming and outgoing data (facilitating data-format independence), abstract internal implementation of the functionalities of components (avoiding invocation dependencies), etc.

Interoperability. An e-contracting system has to be able to interoperate with information systems supporting other business functions (e.g., planning, logistics, production). It has to be able to interoperate with information systems of the external partners (contracting partners, mediators) as well. Integrability and interoperability qualities must be both addressed in terms of data and process aspects.

2.3. Required non-functional architecture qualities

In Bass et al. (2003), a number of general qualities required in architectures are presented. Inspired by them, and after conversations with stakeholders, we have elaborated the following list of architectural qualities expected in ERA:

Completeness. Clearly, completeness is of major importance for ERA, as it has to serve as a guiding model for the design of concrete e-contracting architectures regardless of the business and technological context. A quality closely related to completeness is *scope*. An e-contracting reference architecture must give a clear description of the business aspects that it addresses.

Feasibility. An architecture specification must be implementable (*buildable*), preferably in an easy and timely manner. Furthermore, being a reference architecture, ERA must have a clear *structure* and coherent design (*conceptual integrity*).

Applicability. ERA must be applicable for the design of new e-contracting systems as well as for the analysis of existing systems.

Usability (*acceptability*). For its successful adoption, ERA must be easy to understand by both business and IT professionals (for business professionals to a certain high-level of detail). It must foster communications between the stakeholders of an e-contracting system. Furthermore, architecture designers should be able to use ERA as a starting step in the design of concrete e-contracting architectures. Satisfying these requirements will set the foundations for the acceptance of ERA by both business and IT professionals.

3. Design approach

ERA is designed in accordance with the principle of functional decomposition of a system (also known as "separation of operation"). ERA has three levels of decomposition. At each level of decomposition of ERA, the identified sub-components provide functionalities that are non-overlapping with the remaining sub-components at this level (the "part-whole" principle).

There are a number of qualities that can be addressed through the part-whole decomposition approach. As indicated in Bass et al. (2003), part-whole decomposition promotes achieving *modifiability* and *integrability* qualities in a system. We use functional decomposition also to address the *usability* quality of ERA. High-levels of decomposition can be easily understood by business professionals and used for communication of ideas and opinions, while lower levels of decomposition provide details required by architecture designers. Functional decomposition facilitates the *application* of ERA for analysis of existing e-contracting systems as well. It allows the selection of the proper level of detail for analysis of existing systems. The approach of functional decomposition offers one more benefit. System functionalities can be addressed in a step-by-step manner,

starting from a high level of abstraction and gradually increasing the level of detail. Through a careful control on the decomposition process, the possibility to omit the support of functionalities at a more detailed level is minimized. This helps for achieving *completeness* of ERA at its lower level of detail. However, to ensure completeness of ERA, it has to be controlled for this quality at its first level of decomposition as well. As already mentioned (see Section 2.1), we have addressed functional completeness of ERA by using existing research findings and by conducting discussions with potential stakeholders.

The usage of part-whole decomposition as a leading design choice may have negative effects on performance which we briefly mentioned as a required quality in an e-contracting reference architecture (see Section 2.2.1). This problem has to be addressed in the development view of an e-contracting architecture (Kruchten, 1995, Rozanski and Woods, 2005).

To address the interoperability, security, flexibility, and automation qualities we introduce dedicated components. Conceptually, information systems in an enterprise can be divided into two classes, namely, "external" and "internal" information systems. "External systems" are geared towards communicating data and process specifications between organizations. These systems operate with data and process specifications that are tailored to a commonly agreed data and process semantics. "Internal systems" are geared towards enactment of processes and data management in the context of a specific organization. According to this classification, an e-contracting system can be defined as an external system. An e-contracting system interacts with "internal" information systems. Furthermore, an e-contracting system interacts with the "external" information systems of counterparties. In ERA, clear connections are made through dedicated interfaces and components to partner and internal information systems. This sets the basis for achieving *interoperability* of e-contracting systems. The *security* system quality is addressed in ERA through the introduction of dedicated, security-related components, and the *flexibility* quality is addressed by delegating advanced functionalities ("business intelligence") to certain components. To address the requirement for a *highly automated* system, we design components that can automatically support the required contracting activities.

To achieve *structure* and *conceptual integrity* of ERA (part of the *feasibility* quality), established architectural styles and patterns are used (Bass et al., 2003; Buschmann et al., 1996; Klein and Kazman, 1999). Patterns are used to address the *integrability* quality as well.

In the recent years, Service Oriented Architectures (SOA) (Papazoglou, 2007) gained popularity as a paradigm for software design. SOA provides the foundations for design of architectures of highly-modifiable systems with loosely coupled components. Thus, designing ERA from a SOA perspective can be perceived as a logical step. However, we think that in the case of ERA the design of a carefully engineered, traditional, component-based architecture will offer the benefits introduced by SOA and will avoid a number of negative features that are related to the usage of SOA. Next, we briefly present our argumentation for the advantage of designing a component-based e-contracting architecture over a service oriented e-contracting architecture.

One of the strong features of SOA is that it allows services to be publicly accessible and used. Consequently, applications may use "foreign" services for their own operation. However, relying on "foreign" services or granting them access to services that are part of a critical, private, internal process like contracting is not an option. Thus, this SOA feature is not beneficial in the case of ERA. Another positive characteristic of SOA is the independence of services from their concrete software realization. We abstain from concrete technological choices in ERA as well. As already mentioned, SOA facilitates the design of modifiable, loosely-coupled architecture. By making use of certain styles and patterns, a modifiable architecture of loosely coupled components can be designed as well (Petritsch, 2006).

The usage of SOA for the design of ERA would have a number of weak points. The SOA approach focuses on the description of the interfaces of services and ignores their internal realization (where a service is the software reification of a business function). Thus, using SOA for the description of ERA will lead to a description of a set of coarse-grained e-contracting services without specification of their internal realization. As a result, we would define a highly general architecture, failing in our design objective of defining a detailed reference architecture. SOA is not fully standardized and bad performance and data overhead are issues that still have to be addressed in SOA (Petritsch, 2006). Selection of SOA for the design of ERA will make these current SOA problems an intrinsic characteristic of ERA which may be perceived by users as a limitation of the architecture.

For these reasons, we define ERA as a component-based architecture. We apply styles and patterns to achieve an architecture of modifiable and integratable components. ERA can be mapped to a SOA by selecting modules that provide specific business functions and positioning them in a service oriented architectural framework. Next, in Section 4, we discuss the components from the first level of decomposition of ERA. Components from this level of abstraction represent self-contained business functions and are thus a suitable input for the translation of the ERA components to services in a SOA.

4. First level of decomposition of ERA

As explained in Section 3, ERA is presented at three levels of detail. In this section, first, we discuss the top (first) level of decomposition of ERA. A simple notation consisting of "Components", "Higher-level components", "Passive data components", "Abstract data components", "Information systems of third-parties" and "Data flows"

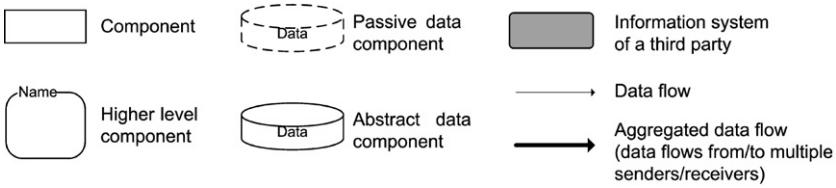

Fig. 1. Notation used for the description of ERA.

(see Fig. 1) has been used in the design of ERA. Passive data components implement the "Data indirection" architectural style (Klein and Kazman, 1999), while abstract data components implement the "Abstract data repository" architectural style (Klein and Kazman, 1999). Next, we discuss loose coupling of the components from the first level of decomposition of ERA and demonstrate that the modifiability and integrability qualities of ERA (see Section 2.2.2) are preserved at this level of decomposition.

4.1. Top-level components and their organization

According to the functional requirements analysis (see Section 2.1), the four e-contracting phases require the support by an e-contracting system of four different, main functionalities. In addition, an e-contracting system must provide support for the management of the e-contracting process. Thus, an e-contracting system can conceptually be decomposed into five general components, i.e., *Matchmaker*, *Partner Selector*, *Contractor*, *Enactor*, and *Contracting Manager* (see Fig. 2). The *Secure Messenger* component is defined in order to address the security quality (see Section 2.2.1). The Secure Messenger component provides support for message encryption and decryption, digital signature management, semantic mapping of messages, and verification of messages for compliance with the process agreed.

The components of the first level of decomposition of ERA are organized in accordance with the layered architectural style (Bass et al., 2003). The Secure Messenger provides communication services to the contracting components (Matchmaker, Partner Selector, etc.). The Contracting Manager uses the services of the contracting components for contract establishment and enactment. In Fig. 2, layers are depicted in reverse order, i.e., lowest layer is depicted on top. The reason for this reverse order is that

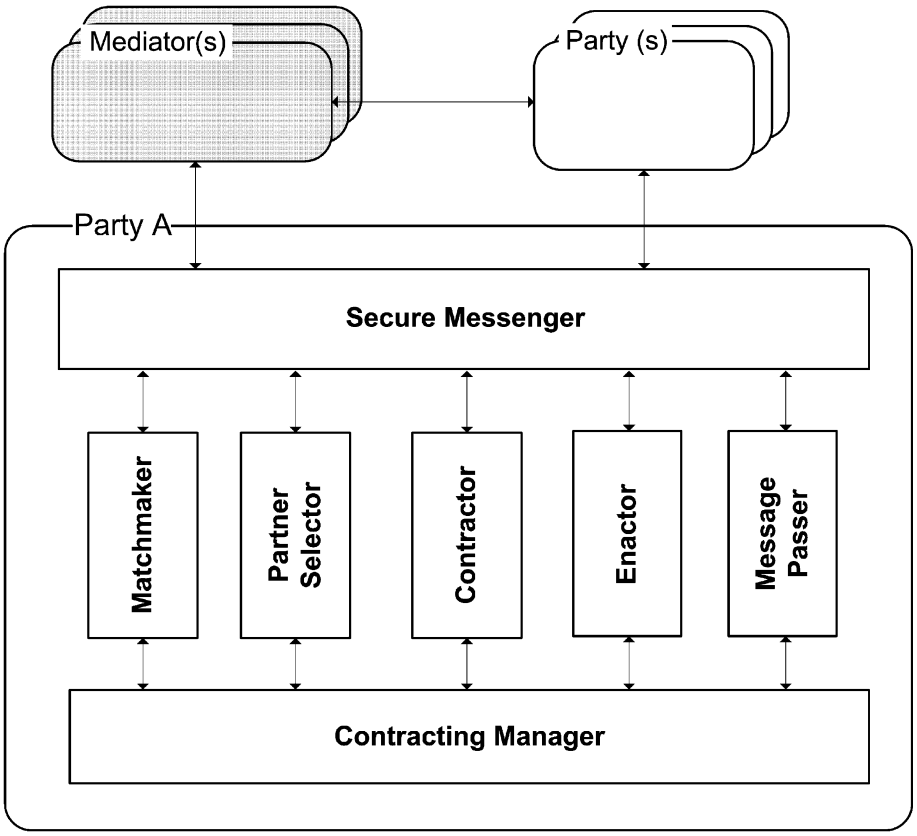

Fig. 2. E-contracting reference architecture (first level of decomposition).

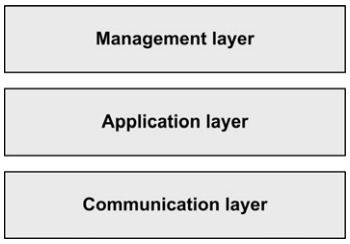

Fig. 3. Layered view of ERA.

people more easily assimilate and accept elements positioned at the top of a picture as external. In Fig. 3, layers are depicted in the traditional order. The Secure Messenger component represents the "Communication layer". The Matchmaker, Partner Selector, Contractor, and Enactor components are part of the "Application layer". The Contracting Manager component represents the "Management layer". The Contracting Manager must exchange certain information with the Secure Messenger (explained in Section 5.1 and 5.5). To avoid direct communications between the Contracting Manager and the Secure Messenger (and thus, to adhere to a strict layered style), we introduce the Message Passer. Its role is to serve as a transition point for messages between the Contracting Manager and the Secure Messenger.

Each component is composed of sub-components. Sub-components in the application and communication layers provide methods that can be invoked by components from the corresponding higher layers. To facilitate integration of components and improve their modifiability, we use the "Facade" pattern (Gamma et al., 1995). This pattern groups the methods offered by sub-components of a component and exposes them for access, while "hiding" details on location, internal naming, etc. Facades allow separate development of components from different layers and their easy integration. To improve integrability between components, we also use the "Abstract data repository" style (Klein and Kazman, 1999). This style allows components to exchange data unaware of each others requirements on the data format.

As discussed in Section 2.1, in a limited set of business scenarios, the matchmaking and partner selection functionalities are not required. In these cases, the Matchmaker and Partner Selector components can be omitted in the design of concrete e-contracting systems.

The components from the first level of decomposition of ERA are defined on the basis of a set of non-overlapping, self-contained functionalities. This ensures loose functional coupling of the components from this level. However, it does not ensure that the components from this first level of decomposition are actually loosely coupled from a data perspective. To address this problem, next, we discuss the collaborations in which the components from the first level of decomposition of ERA are involved. To describe the collaboration, we define a structural model and a behavioral model (Booh et al., 1999). The structural model describes the data entities created, read, updated, and deleted by the components, while the behavioral model describes the way components manage these data entities (creating them, distributing them in the form of messages, etc.). We use these models to show that the components from the first level of decomposition of ERA are loosely coupled from a data perspective as well. These two models can be seen as part of the information view in Rozanski and Woods (2005).

4.2. Collaboration model

The components from the first level of decomposition of ERA participate in two high-level collaborations. The "contract establishment" collaboration aims at the establishment of a contract. It involves all high-level components except the Enactor. The "contract enactment" collaboration is a collaboration between the Contracting Manager, Enactor and Secure Messenger components and is organized towards the enactment of a contract. As in both collaborations, the role of the Secure Messenger is of minor importance, we do not further pay explicit attention to it.

In this section, we describe only the "contract establishment" collaboration. The "contract enactment" collaboration involves a small number of interactions between the Contracting Manager and Enactor components. The Contracting Manager can send a message to the Enactor to initiate the enactment of a contract (which is included in the message) and optionally can prematurely terminate the enactment of the contract due to internal factors (see Section 5.4). As the structural and behavioral aspects of the "contract enactment" collaboration are relatively simple, the lack of data dependencies in it can be directly observed. For this reason, we do not further discuss it.

To avoid unnecessary complexity, we omit minor details in the description of the "contract establishment" collaboration and concentrate on its core aspects. We use UML static structure and collaboration diagrams for the description of the structural and behavioral models, respectively (Booh et al., 1999).

4.2.1. Structural model

A "contract request" (see Fig. 4) is an internally generated message that expresses the need within the company of a contract for a certain exchange value (a product, service), in certain quantity, and with certain qualities. A request for matchmaking ("matchmaking request") is based on the contract request and contains in addition a number of matchmaking rules. A rule may state what the geographical location of potential partners must be, may define operational constraints (e.g., time for collecting of matching parties), etc. An advertisement ("own advertisement") is created on the basis of the matchmaking request. It contains a looser version of the qualities and quantities defined in the matchmaking request (in order to broaden the search for partners). For simplicity, we consider queries requesting

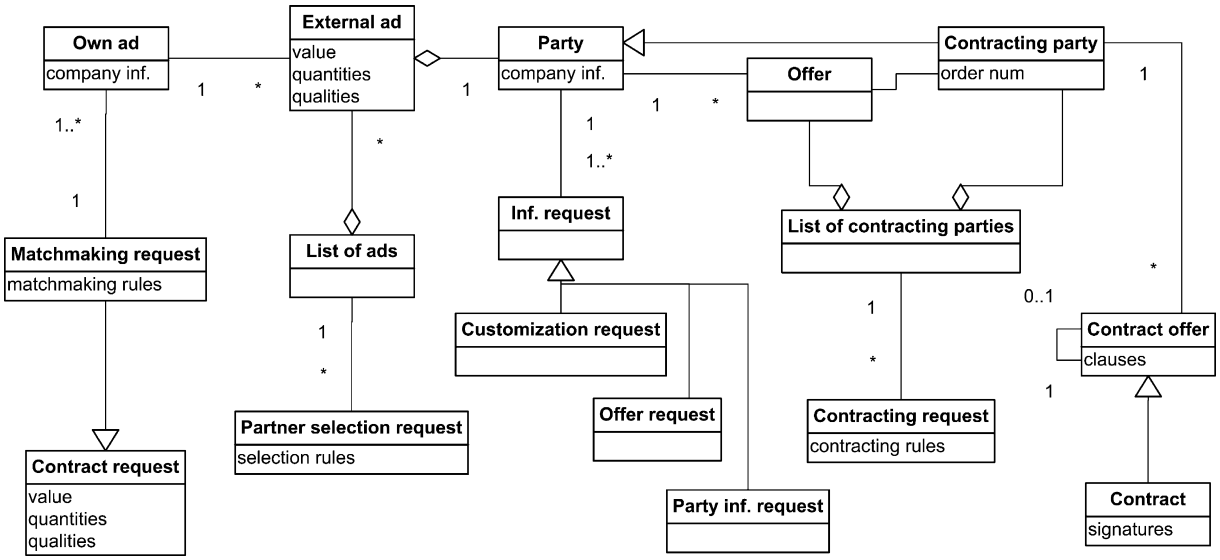

Fig. 4. Structural model of the "contract establishment" collaboration.

products or services that are sent to other parties also as advertisements. Own advertisements also contain information for the company. For each own advertisement, matching advertisements of other companies may be found ("external advertisement"). A matching advertisement contains information for the advertisement and information for the party behind this advertisement. All matching advertisements form a "list of matching advertisements".

The "partner selection request" is a message that triggers the selection of the most suitable partner for negotiations. It is created on the basis of the list of matching advertisements and on a number of rules that influence the selection of a party. For example, a rule can state that for the selection of a partner time of value delivery is more critical than price of the value. Additional information for the business status of the parties is collected through requests for information ("information request"). These requests can be for an offer (an advertisement does not have the legal status of an offer and may fail to provide important information), customization of an offer, or additional information for the party itself. An offer is a legally binding document that lies in the foundations of the establishment of a contract. An offer would usually lack many legal and business clauses that are added later on in contracts. A "contracting party" is a party that is selected as a potentially "good" party for negotiation. A contracting party has a ranking as a partner compared to the other contracting parties (i.e., it shows how "preferred" is this party for the establishment of a contract). All contracting parties and their final offers form a "list of contracting parties".

A "contracting request" for the start of contract negotiations is created on the basis of the list of contracting parties and on a number of rules that should influence the contract establishment process (e.g., "establish a contract until..."). Contract offers are exchanged with contracting parties. A contract offer that is accepted by the negotiating parties is signed and becomes a contract.

4.2.2. Behavioral model

To simplify the behavioral model, we depict it into three separate diagrams. First, the *Contracting Manager* (CM) component receives a *contract request* for the establishment of a contract (see Fig. 5). The CM elaborates on the basis of the internal request a *matchmaking request* that is sent to the *Matchmaker*. On the basis of the *matchmaking request*, the *Matchmaker* creates *own advertisements* and publishes them at external parties. The *Matchmaker* receives *external advertisements* that match its own advertisements. It elabo-

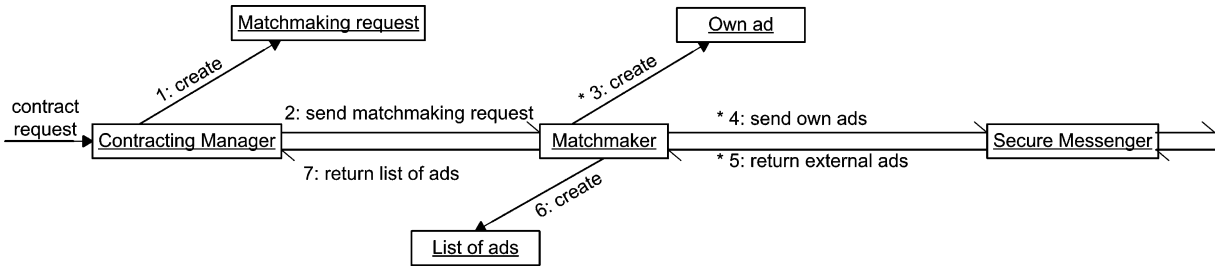

Fig. 5. First part of the contract establishment collaboration.

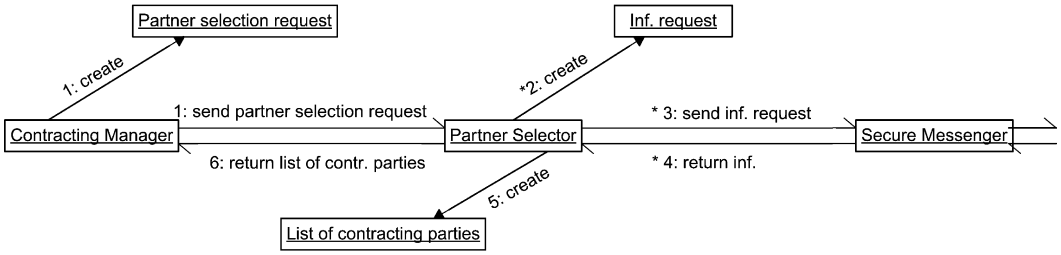

Fig. 6. Second part of the contract establishment collaboration.

rates a *list of matching advertisements* and returns it to the CM.

The CM evaluates the result (e.g., length of the list). Dissatisfactory results may lead to a new invocation of the Matchmaker (not shown in this collaboration scenario). If the result is satisfactory, it creates a *partner selection request* and sends it to the *Partner Selector* (PS) (see Fig. 6). The PS creates a number of *information requests* which sends to parties (including third parties) and on the basis of the answers received creates a *list of contracting parties* which returns to the CM.

If the *list of contracting parties* is not empty, the CM uses it to define a *contracting request* which is sent to the *Contractor* (see Fig. 7). The *Contractor* creates *contract offers* and if an agreements is reached with the counter party, creates a *contract*. The contract is returned to the CM.

A stepwise analysis of this collaboration shows that each data entity from the structural model is produced by a component and is used by a component. Each component receives the data that it needs for its operation and produces the data required by the next component in the collaboration in a sequential manner. Thus, there is both functional and data decoupling among components from the first level of decomposition of ERA.

Next, in Sections 5 and 6, we provide descriptions of the second and third levels of detail of ERA, respectively. For brevity reasons, we do not provide structural and behavioral models in the second and third levels of decomposition of ERA. Rather, we depict the main messages that the components exchange and discuss them in the text description. This allows readers to form a limited idea for the level of data coupling among sub-components.

5. Second level of decomposition of ERA

This section presents the decomposition of the Secure Messenger, Contracting Manager, Partner Selector, Contractor, and Enactor components.

The functionalities provided by the Matchmaker have been researched for many years and research findings on and implementations of this component are available (Bichler and Segev, 1999). For this reason, the Matchmaker component is not further decomposed in this paper. The Matchmaker can be implemented by a company as an application with complete functionality. Alternatively, its main functionalities (matching of parties based on their offers/requests) can be outsourced to an external third party. The company has to implement in this case only a light version of the Matchmaker that supports sending and receiving of current offers/requests to and from an external Matchmaker. A heterogeneous solution that implements both alternatives is possible as well. An example of such solution is discussed in Section 7.2.1.

The Message Passer was introduced to serve pure architectural goals. Its functionalities do not require further attention.

The decomposition of the Contracting Manager is based on existing reference architectures in workflow management. The Partner Selector, Contractor and Enactor components provide support for the pre-contracting, contracting, and enactment phases of the e-contracting process, respectively. That is why we use the e-contracting process decomposition (see Section 2.1) for their functional decomposition. The decomposition of the Secure Messenger is based on the functional requirements on this component defined in Section 4.

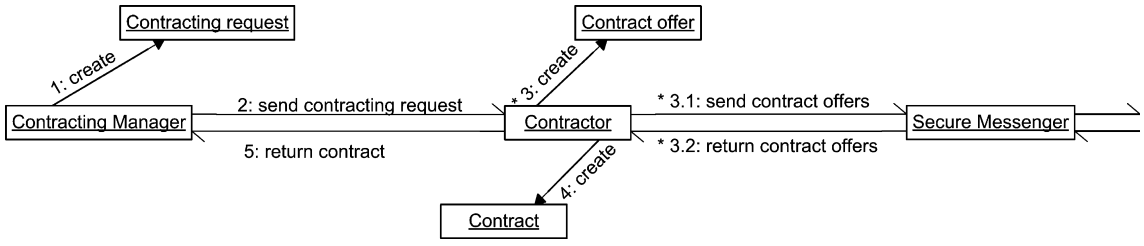

Fig. 7. Third part of the contract establishment collaboration.

5.1. Decomposition of the Contracting Manager component

The Contracting Manager component provides support for the management of the e-contracting process. It can be seen as a special kind of a workflow management system. For this reason, the high-level view of the Reference Architecture for Workflow Management Systems (Grefen and Remmerts de Vries, 1998) abbreviated here as RAWFMS, and the Workflow Reference Model (Hollingsworth, 1995) abbreviate as WRM are used as an inspiration for the decomposition of the Contracting Manager. The two workflow reference architectures are aligned with the requirements defined on ERA and are tailored for the domain of e-contracting. As a result the Enactment Server, Control User Interface, Definition User Interface, Design, Evaluator, Knowledge Updater, Internal Mapper, and Internal Broker sub-components are defined (see Fig. 8). Next, each of these sub-components, their relation to the RAWFMS and WRM, and their specifics for the domain of e-contracting are described.

5.1.1. Definition User Interface (DUI) and Design components

The DUI component provides the user interface for the specification of supported e-contracting activities and the rules applying to these activities. Furthermore, this component allows users to define contract templates and clauses that can be used during contract negotiation. The Design component supports the specification process and controls the user input for consistency with existing definitions. The DUI and Design components have similar functionalities to the "UIS interface" and "WF Design" components in RAWFMS and the "Process Definition Tools" component in WRM.

5.1.2. Knowledge Updater (KU) component

The Knowledge Updater collects/receives relevant information from other information systems. New business rules, process specifications, service/product specifications defined in internal company systems that are relevant for the contracting process are sent to/requested by the KU. For example, the process supported by an internal system for delivering of a service (named Internal Enactor in Fig. 8) can be used in the specification of the service in a contract (Grefen et al., 2000). The KU obtains relevant information from external systems as well (e.g., currency rates, new contract templates). Information collected by the KU may be controlled and approved via the DUI and Design components.

5.1.3. Enactment Server component

The Enactment Server component, triggered by the Internal Planner, starts and manages e-contracting processes. The Internal Planner belongs to the internal information systems (shown in grey color) of the company (see Section 3) and provides information for required or available for contracting products/services. In the manufacturing domain, an application that serves as an Internal Planner is the Material Requirements Planning (MRP) application. MRP modules are implemented in leading ERP systems like SAP and Oracle (Nahmias, 1997). Based on the process specifications and rules and on the data produced by already invoked components, the Enactment Server invokes or terminates the execution of components.

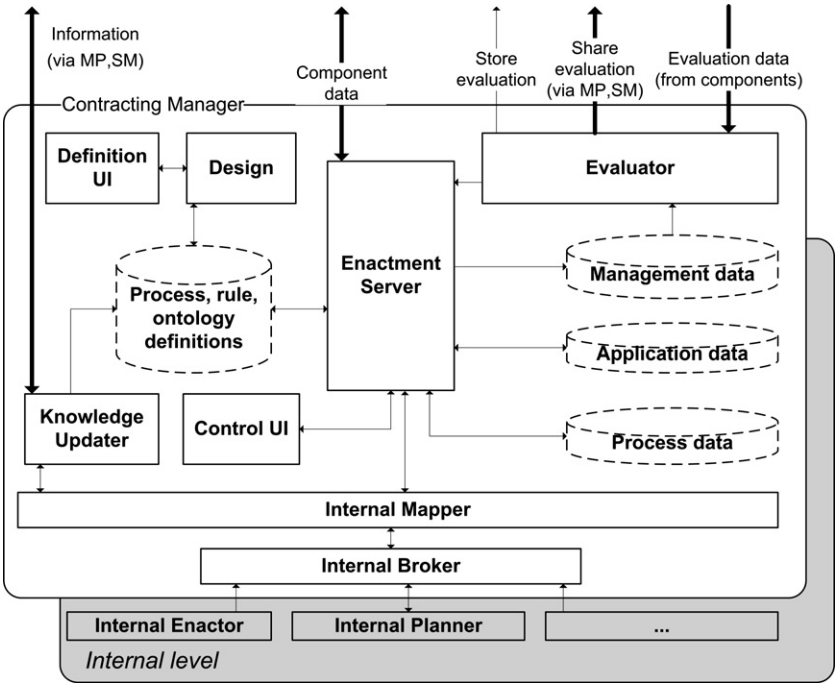

Fig. 8. Decomposition of the Contracting Manager component.

The status of a contracting process is stored in "Process data" database. When invoking a component, the Enactment Server supplies configuration data to the invoked components. The configuration data is produced during the execution of other components (stored in the "Application data" database) or is defined before the initiation of the contracting process (stored in the "Process, rule, ontology" database). The configuration data for the Communication Monitor contains the specification of the contracting and enactment process (see Section 5.5). The configuration data for the Partner Selector is a list of matching partners found by the Matchmaker. Similarly, the relevant configuration information is sent to the other e-contracting components as well. In order not to overburden the figures, channels for configuration information are not shown. The "Management data" database contains aggregated workflow process data and management information that is used by the Evaluator for evaluation purposes (explained next).

The Enactment Server component has functionalities similar to the "Workflow Engine" component in the WRM and the "WF Server" component in the RAWFMS. However, in contrast to existing workflow engines, the Enactment Server should be able to monitor and, eventually, control relations between contracting process instances. This is required as contracting processes are often related to each other. These relations are also known as contracting dependencies (Angelov and Grefen, 2001). Failure in one contracting process may lead for example to the need of termination of other contracting processes. This characteristic of the Enactment Server requires specific attention in its design.

5.1.4. Evaluator component

The Evaluator component facilitates the contract-management support provided by the Enactment Server. Many architectural components can provide information that is valuable for the evaluation of the e-contracting relationship. For example, a high number of invalid digital signatures reported by the Authenticator might indicate higher risk of fraudulent attempts (see Section 5.5). Lack of offers that match the needs of the company for a long period of time would give a sign for possible contracting process failure or at least delay of contract establishment. Frequent occurrence of contract disputes may be indicative for bad communication channels, low contract quality, or simply for an unreliable partner. The Evaluator collects statistical information stored by other components and using predefined algorithms provides an evaluation statement on the contracting relationship at a specific point in time. An algorithm can state which statistical information must be used, assign different weights on it, etc. The information that can be collected and the algorithms for its evaluation require further research. In order not to overburden the figures representing ERA, the communication channels through which evaluations are obtained are not shown. Based on the information provided by the Evaluator, the Enactment Server can change the contracting process (terminate it, suspend it, etc.). At the end of the contract enactment, the Evaluator produces a final evaluation of the e-contracting relationship. This final evaluation is to be used for adapting the future contracting behavior of the company. For example, it can be used by the Partner Selector in the future selection of partners (see Section 5.2). An evaluation can be stored externally at a Reputation Ranking Center (Masum and Zhang, 2004) as well. The final evaluation activity is part of the enactment phase (see Table 1). However, due to the need for collection of intermediate evaluations throughout the complete e-contracting process and for sharing these evaluations with the Enactment Server, the Evaluator component is defined as a sub-component of the Contracting Manager.

The Evaluator component can be seen as a concretization of the "Application Systems" component in RAWFMS. The Evaluator component is part of the "Workflow Engine" in WRM.

5.1.5. Control User Interface (CUI) component

Similar to the "Administration and Monitoring Tools" component of the Workflow Reference Model, the CUI provides access for humans to monitor and control the Enactment Server. This user interface allows handling of potential exceptions that occur in a contracting process and require human intervention. The analogue of CUI in the RAWFMS is the "Workflow administration client".

5.1.6. Internal Mapper and Internal Broker components

The Internal Mapper facilitates the integration of the Contracting Manager with internal systems. It maps method calls and data formats of internal systems to the data format and methods used in the Contracting Manager. Thus, any changes in the internal systems are reflected only on the Internal Mapper. The Internal Broker is responsible for delivering messages between the Contracting Manager and internal systems. It implements the Broker pattern (Buschmann et al., 1996) and provides location and implementation transparency of internal systems.

The RAWFMS provides a detailed decomposition of its components and can be used for the further decomposition of the components of the Contracting Manager.

5.2. Decomposition of the Partner Selector component

The Partner Selector component supports the pre-contracting phase. As shown in Table 1, it must be able to collect external information about potential parties (see "Offer" and "Partner information" abstract activities). Based on the collected external information and on existing internal information, the Partner Selector has to select partners with which negotiations on a contract may start. To support the information collection activities we define the Information Collector sub-component of the Partner

Selector. To support the selection activity, we define the Selector sub-component (see Fig. 9).

5.2.1. Management data (PS), Evaluations, Selection rules data components

In the "Management data (PS)" database, data is stored that is required for the performance of the Partner Selector and that is provided by the Contracting Manager (e.g., parties to be considered). Evaluations from previous business relations are stored by the Contracting Manager in the Evaluations database. These two databases implement the "Abstract data repository". style to reduce coupling between the Partner Selector and Contracting Manager components (possibly developed in isolation). The Selection Rules database contains the rules for selection (defined via the Definition UI).

5.2.2. Selector component

The Selector component obtains the set of potential partners (discovered by the Matchmaker) from the "Management Data" database. Based on evaluations stored in the Evaluations database (see Section 5.1), the internal rules, and on the information provided by the Matchmaker, the Selector attempts to nominate a "winning" partner (or a set of "winners"). If the information is not sufficient or not up-to-date, the Selector requests the Information Collector for collection of additional information. The Selector implements functionalities that are typical for a Decision Support System. Decision Support Systems have received sufficient attention in the research world and a number of commercial systems for development of rule-based applications are available (e.g. ILOG, 2006; Savvion, 2006). Therefore, the Selector component is not further decomposed in this paper.

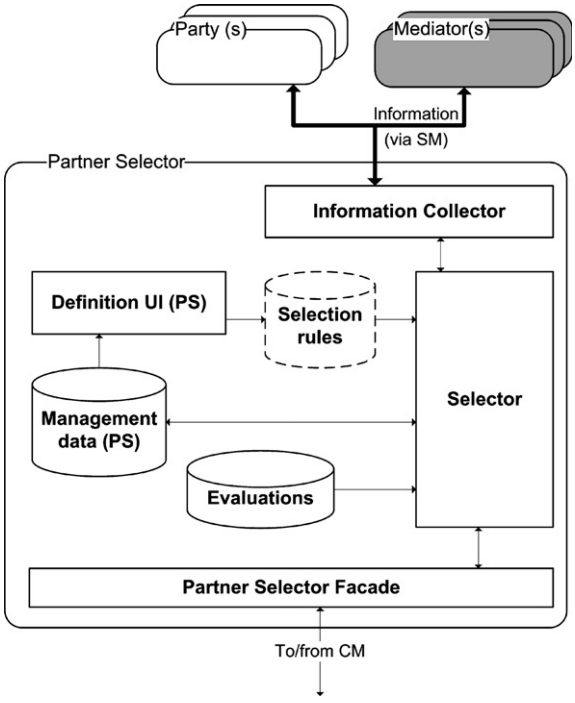

Fig. 9. Decomposition of the Partner Selector component.

5.2.3. Information Collector component

In many scenarios, the information available internally and the information discovered by the Matchmaker may be insufficient for nominating a "winning" partner. For example, the Matchmaker might have identified a matching offer but with no price information. Furthermore, guarantees, credentials, up-to-date offers, general provisions, etc. may be required for the choice of a partner with whom to start negotiating a contract. The Information Collector is responsible for collecting additional information for a specific partner. The information may be obtained from Trusted Third Parties (e.g., chambers of commerce), trusted business partners, or the potential partner-company itself. In certain business domains, the Information Collector may stage a tender auction for collecting of offers from potential partners. The Matchmaker component is responsible for providing information on requests arriving from the Information Collectors of other parties.

5.2.4. Definition UI (PS) component

The Definition User Interface of the Partner Selector allows users to define the rules for the selection of the potential partners. It uses data from the Management data (PS) on the offered/required products and services for the rule definitions (e.g., for a certain product speed may be of higher importance than costs).

5.3. Decomposition of the Contractor component

The Contractor component supports the execution of the activities in the contracting phase. In Table 1, the contracting phase is decomposed into two abstract activities, i.e., "Negotiation", and "Signing and storing". Consequently, we define the Negotiator and Contract Finalizer sub-components to support these sub-activities (see Fig. 10).

5.3.1. Negotiator component

The Negotiator component supports negotiations on the values to be exchanged, the e-contract content, non-agreed updates of e-contracts, or other negotiable topics. The Negotiator has access to a set of rules that define the company policies for negotiation, contract templates, etc. If the Partner Selector provides a set of "partner candidates", the Negotiator will have to start multiple negotiations threats, looking for the best trading opportunity. Depending on the context, the Negotiator component may have to implement a complex functionality that will often require high degree of flexibility. In Beam and Segev (1997), a comprehensive survey on the existing research results on this topic is presented. In Jennings et al. (2000), an architecture of a negotiation component named Interaction Management Module (IMM) is presented. It can be used for further decomposition of the Negotiator component.

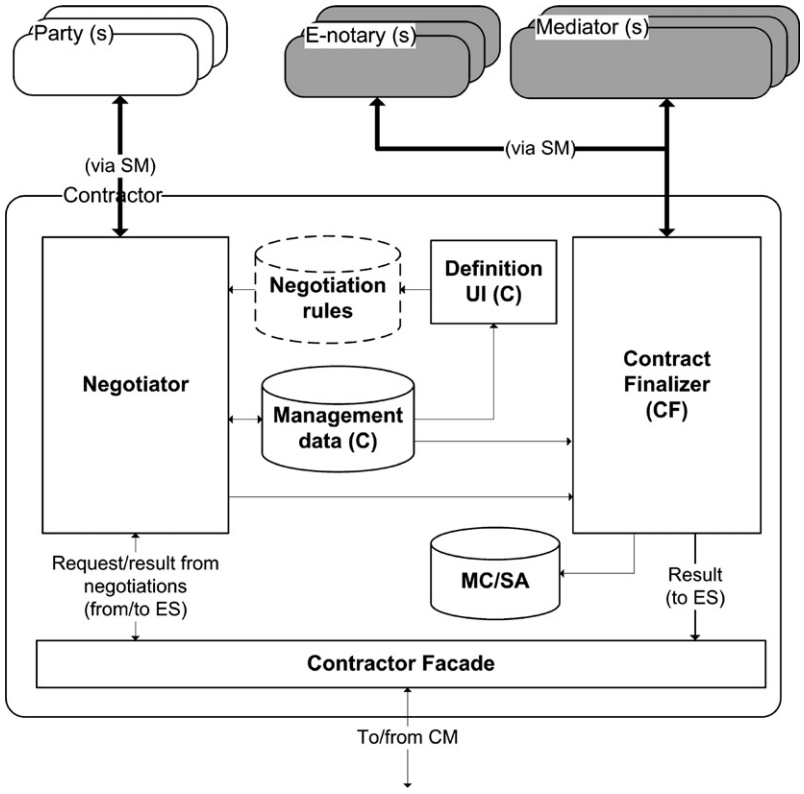

Fig. 10. Decomposition of the Contractor component.

5.3.2. Contract Finalizer (CF) component

When an e-contract (or an update of an e-contract) is agreed upon, the Contract Finalizer arranges the establishment of the e-contract and its storing. It implements a mechanism that guarantees the signing of the agreed e-contract by all parties (e.g., via an e-notary). When the e-contract is signed by all parties and all parties possess a copy of the e-contract, the CF stores the e-contract internally (in the MC/SA component) and distributes it to the mediators involved in the contract enactment, e.g., to contract enactment monitors, arbitrators, trusted updaters (Angelov et al., 2005). At the end, it informs the Contracting Manager for the final result.

5.3.3. Definition UI (C) component

The Definition User Interface (C) allows users to define negotiation rules and strategies for the establishment of an e-contract. It uses the clauses and rules defined via the DUI of the Contracting Manager and stored in the Management data (C).

5.3.4. MC/SA, Negotiation rules, Management data (C) components

In the MC/SA database, original contracts (named *M*aster *C*ontracts) or their updates (named *S*ubsidiary *A*rrangements) are stored. The "Negotiation rules" and "Management data (C)" databases provide data storage functionalities analogous to the databases discussed in the Partner Selector component (see Section 5.2).

5.4. Decomposition of the Enactor component

In the decomposition of the e-contracting process model presented in Table 1, four abstract activities during e-contract enactment are identified, i.e., value exchange, monitoring and control, dispute resolution, and evaluation activities. The support of the evaluation activity is provided by the Evaluator sub-component in the Contracting Manager. To support the monitoring and control activities, the External Enactment Server and Data Manager sub-components of the Enactor component are defined. To support the dispute resolution activities the Dispute Handler is defined. The support for the value exchange activity is provided by the Internal Enactor component that is part of the "internal" enterprise information systems (shown in grey color in Fig. 11). The Internal Enactor represents the information system that supports the execution of business processes related to the value delivery agreed in the e-contract. The Internal Enactor can already be in place before the introduction of an e-contracting system in a company (an example of an Internal Enactor is a workflow management system). As in the Contracting Manager, the Enactment Mapper and Internal Broker components address the interoperability quality (see Section 2.2).

5.4.1. Data Manager (DM) component

The Data Manager component supports the management of contract related data during e-contract enactment. In an e-contract, parties agree on certain rules for updating

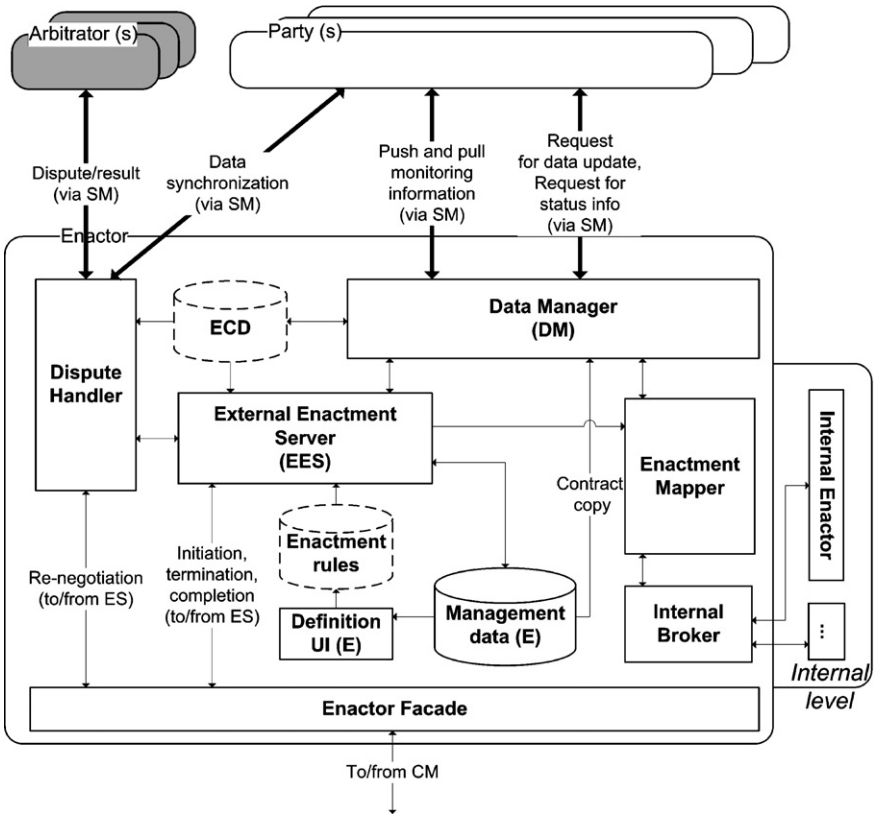

Fig. 11. Decomposition of the Enactor component.

some of the contract data and for exchanging its new values (Angelov et al., 2005). During e-contract establishment, data items that may have their values updated are assigned initial values that are agreed upon by the parties at the time of contract establishment (could be a null value as well). We denote this type of data as External Contract Data items (ECD). As the new values of ECD items are not part of the agreement, these new values are stored externally of the contract (hence their name). An example for an ECD item is an "event data item". In the e-contract, the parties agree the occurrence of the event to be reported by one of the parties. Upon its occurrence, the responsible party informs the counterparty that the event data item status has changed from "not-occurred" to "occurred". When a request for an update of an ECD occurs, the Data Manager verifies if the requester is allowed to update the value of this data element. If the verification result is positive, the Data Manager updates the value and informs the Internal Enactor (via the Enactment Mapper and Internal Broker) and if required – the counterparty (push monitoring). On request from the counterparty, the Data Manager can provide information on the current values of the ECD (pull monitoring).

5.4.2. External Enactment Server (EES) component

The External Enactment Server component manages the e-contract enactment. Based on the interpretation of the e-contract, current state of the enactment process and ECD, and the defined internal business rules, it schedules and starts the execution of activities agreed in the e-contract. In case a contract violation is detected and the EES estimates that the violation must be addressed, it sends a request to the Dispute Handler (the EES might purposely postpone or omit a call for contract violation). To maintain high level of awareness, the EES can request information from the counterparty on the current value of certain data items (through the DM). The EES can make a decision to purposely violate the e-contract enactment (when the benefits will be higher than the losses), to terminate the e-contract enactment, or to request re-negotiation of the e-contract. When one of these decisions is taken, the Contracting Manager is informed. Clearly, the EES must support rather complex (in terms of business intelligence) functionalities.

The Enactment Server manages the global contracting process (see Section 5.1). As the ES has a global view over the contractual relationships of a company, it has higher level of awareness and may instruct the EES to terminate/suspend/resume the contract enactment even if the EES does not estimate this as necessary.

For decomposition of the EES, two components, part of an agent-based contracting architecture presented in Jennings et al. (2000) can be used. The first component is called Service Execution Module (SEM) and provides support for triggering, suspending, resuming, and terminating activities agreed in the e-contract. It handles

"low-level exceptions" as well (e.g., attempting corrections by restarting a failed task). The second component is called Situation Assessment Module (SAM) and provides support for monitoring the state of an e-contract and selecting and scheduling of activities to be performed (communicated to SEM). SAM must be extended to support collection of data from the counterparties. SAM may be extended with a user interface for manual handling of exceptions.

5.4.3. Dispute Handler component

The Dispute Handler component provides support for handling disagreements that have occurred during the e-contract enactment. Based on the information provided by the External Enactment Server, it attempts to identify the cause for the dispute and to resolve it. If re-negotiation is necessary, the Contracting Manager is informed. If resolution is not possible, an external Arbitrator is informed. The result of the dispute resolution is returned to the External Enactment Server.

5.4.4. ECD, Enactment rules, Management data (E) components

The "ECD" database contains the past and current values of the External Contract Data items. The "Enactment rules" and "Management data (E)" databases provide data storage functionalities analogous to the databases discussed in the Partner Selector component (see Section 5.2).

5.4.5. Enactment Mapper, Internal Broker components

The Enactment Mapper component is used to address the interoperability system quality discussed in Section 2.2. The Enactment Mapper component provides "passive" and "intelligent" mapping between data and process specifications used in the Enactor and data and process specifications used at the internal level. Passive mapping is a one-to-one mapping between two data/process items that have different names at the internal and external levels. Intelligent mapping is used for mapping of one data/process item from the external level to several data/process items from the internal level. Intelligent mapping might require processing of the mapped data. Rules for applying intelligent mapping techniques on process specifications are discussed in (Zdravkovic and Kabilan, 2005). The Enactment Mapper receives the data intended for the Internal Enactor from the Data Manager (or from the EES in the cases of process invocation) and delivers the data provided by the Internal Enactor back to the Data Manager. Besides, the Internal Enactor, other internal systems may require data from the Enactor. For example, a company may have a financial system that can provide information for received/performed payment related to a contract. To improve decoupling between the

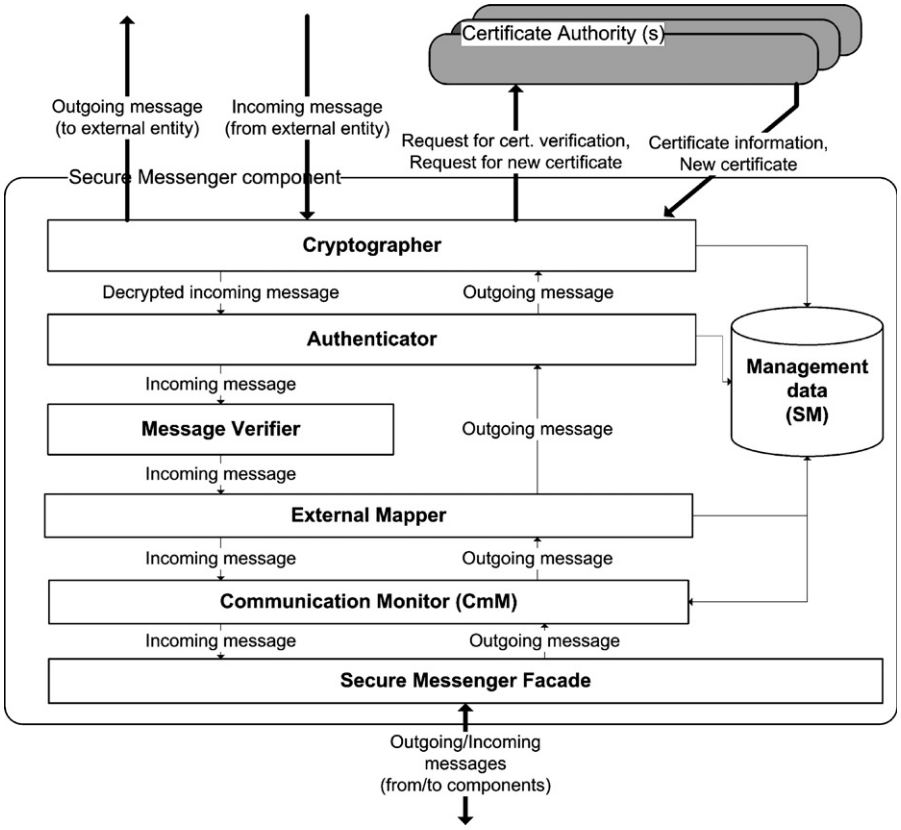

Fig. 12. Decomposition of the Secure Messenger component.

enactor and internal systems, we introduce the Internal Broker (analogously to Section 5.1).

5.4.6. Definition UI (E) component

The Definition User Interface for the Enactor allows users to define rules for the contract enactment (e.g., rules stating when a contract may be violated, rules for tolerance of partner misbehavior).

5.5. Decomposition of the Secure Messenger component

As defined in Section 4, the Secure Messenger must provide five basic functionalities, i.e., encryption/decryption (to address the "encryption" security quality discussed in Section 2.2), sender authentication (to address the "authentication" and "non-repudiation" security qualities), message integrity verification (to address the "integrity" security quality), mapping, and process monitoring functionalities. Consequently, five sub-components of the Secure Messenger are defined, i.e., the Cryptographer, Authenticator, Message Verifier, External Mapper, and Communication Monitor components (see Fig. 12).

5.5.1. Cryptographer component

The Cryptographer receives all incoming messages. If an incoming message is encrypted, the Cryptographer decrypts it. Outgoing messages (sent by the Authenticator) are encrypted by the Cryptographer (if necessary) and sent to the external recipient. Messages that cannot be decrypted are stored in the "Management data (SM)" database. This information is to be used by the Evaluator subcomponent (discussed in Section 5.1).

5.5.2. Authenticator component

The Authenticator component authenticates the identity of message senders. The authentication method should also guarantee non-repudiation from the message authorship. For outgoing messages, the Authenticator applies techniques to ensure that receivers will be able to authenticate (in a non-repudiate way) the company as the sender. The Authenticator may request information from external parties that will allow it to verify sender's authenticity (e.g., a certificates from a Certificate Authority). Information for non-authenticated messages is stored in the "Management data (SM)" database for evaluation purposes.

5.5.3. Message Verifier component

The Message Verifier checks incoming messages for integrity. Information for messages that fail the verification process is stored in the "Management data (SM)" database for evaluation purposes.

5.5.4. External Mapper component

Contracting parties may use different message schemas for their messages (e.g., a "request for offer" message can be described as a "request for quotation" as well). The External Mapper performs bidirectional mapping of the descriptions used by counterparties and the descriptions used by the e-contracting system.

5.5.5. Communication Monitor (CmM) component

In Angelov and Grefen (2004a), it is shown that e-contracting processes can be defined in a fixed, structured manner or in non-structured, rule-based manner (allowing greater flexibility to companies). In both scenarios, there is a specification of the communication activities that can be performed by the parties at any moment.

The Communication Monitor component verifies if incoming and outgoing messages comply with the agreed e-contracting communication process and applicable rules. If a message is disapproved, the CmM stores information about the sender and the rejected message (information to be used by the Evaluator in the Contracting Manager). Outgoing messages from any component of the e-contracting architecture are sent to the CmM and if approved are forwarded to the External Mapper. The CmM keeps the Contracting Manager informed of the current state of the communication activities performed by the parties. The CmM obtains the process and rule specifications provided by the Contracting Manager via the Management data (SM) database.

The sub-components of the Secure Messenger operate in a sequential mode. For each incoming message, a component receives data, processes it, and sends the resulting data to the next component in a strict, unidirectional sequence. For each outgoing message, the components (except for the Message Verifier) again perform their functionalities sequentially but in a reversed order. For this reason, as shown in Fig. 12, the decomposition of the Secure Messenger follows the "Bidirectional batch sequential" architectural style (Bass et al., 2003). The identified sub-components serve as passive filters, i.e., the data is pushed to a component by its preceding component. The batch-sequential style can affect system performance (Bass et al., 2003). This problem has to be addressed in the concurrency view (Rozanski and Woods, 2005) of an e-contracting architecture (e.g., by implementing the "concurrent pipelines" style (Klein and Kazman, 1999)).

6. Third level of decomposition of ERA

This section contains description of the decomposition of a number of components of ERA at the third level of detail. As already discussed, for many components at the second level of detail, either existing design patterns can be used for their implementation or existing reference architectures can be used for their further decomposition. For example, the Enactment Server can be further decomposed based on the "WF Server" component in the Reference Architecture for Workflow Management Systems (see Section 5.1). That is why, next, decomposition solely of the Contract Finalizer, Dispute Handler, and Data Manager components is provided.

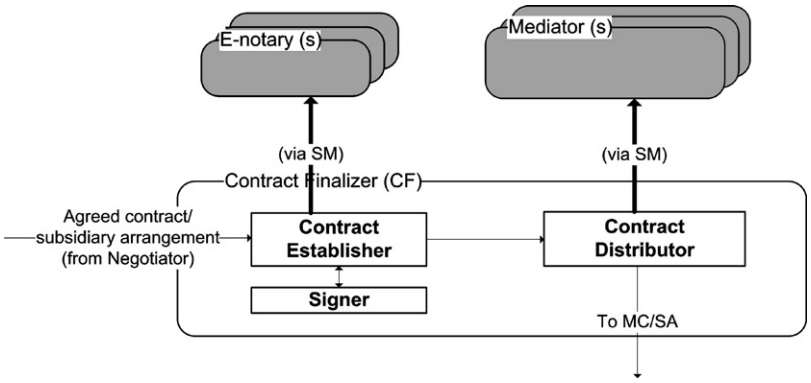

Fig. 13. Decomposition of the Contract Finalizer component.

6.1. Decomposition of the Contract Finalizer component

As already explained, the Contract Finalizer supports the signing of the agreed contract, its storing and optionally distribution to other mediators. Hence, the Contract Finalizer is decomposed into Contract Establisher, Signer, and Contract Distributor components (see Fig. 13).

6.1.1. Contract Establisher (CE) component

The Contract Establisher receives from the Negotiator an e-contract (or a subsidiary arrangement) that was agreed upon by the parties. The Contract Establisher implements a mechanism that guarantees the signing of the e-contract by all parties. An example mechanism that makes use of an e-notary is discussed in Angelov et al. (2005). For the signing of the e-contract, the CE can make use of a Signer component (discussed next).

6.1.2. Signer component

The Signer signs agreed e-contracts. Digital signatures are a common technique for signing of electronic documents. The signing of an e-contract means for a company acceptance of certain obligations. This makes contract establishment a highly important activity from a security point of view.

6.1.3. Contract Distributor component

The Contract Distributor receives from the Contract Establisher an e-contract signed by all parties. The Contract Distributor stores the e-contract in the MC/SA data component and can optionally distribute it to mediators that are involved in the e-contract enactment. The mediators that must obtain a copy of the e-contract are provided by the Negotiator and/or the Contracting Manager.

6.2. Decomposition of the Dispute Handler component

As discussed in Milosevic et al. (2002), resolution of disputes can be automated to a certain extend. The Dispute Handler component provides support for handling of occurring disputes between contracting parties. Based on current contracting practices (Angelov, 2007a), and

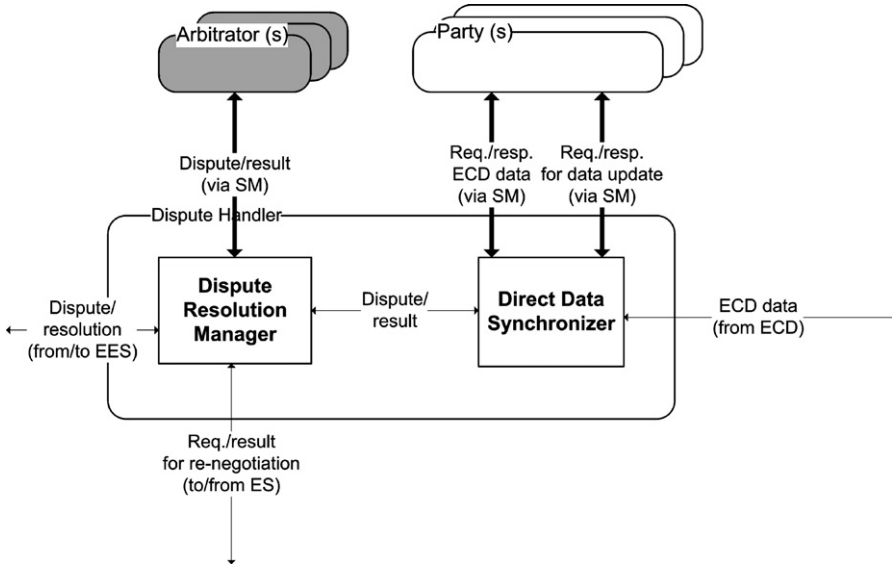

Fig. 14. Decomposition of the Dispute Handler component.

research results for automated dispute resolution (Milosevic et al., 2002) two subcomponents of the Dispute Handler can be defined, i.e., the Dispute Resolution Manager and the Direct Data Synchronizer (see Fig. 14).

6.2.1. Dispute Resolution Manager (DRM) component

The Dispute Resolution Manager component manages the resolution of e-contract disputes. The Dispute Resolution Manager resolves disputes in an escalation manner. When it receives information for a dispute from the External Enactment Server component, first, it attempts resolution of the conflict by comparing the ECD available at the e-contracting parties. The reason for the dispute might be for example a failed update of local data or not sent/received information for data update. For this step, it invokes the Direct Data Synchronizer component (explained below). If there is no synchronization problem or there is no agreement on the synchronization, the DRM requests resolution of the dispute through negotiations (to be performed by the Negotiator component). The DRM sends a request for negotiations to the Contracting Manager and waits for the result. If the Negotiator fails, the DRM provides the dispute case to an external Arbitrator that enforces a resolution of the dispute on the contracting parties. Depending on the situation, the DRM can skip the first and/or second steps in the dispute resolution process. The result of the dispute resolution is returned to the External Enactment Server component.

6.2.2. Direct Data Synchronizer (DDS) component

The Direct Data Synchronizer performs comparison of the ECD available locally and at the counterparty. First, it requests from the counterparty its ECD data. Next, it compares the received data and the local ECD data. If differences are found, the DDS sends a request for update of the differing data items to the counterparty. The request for update contains the differing ECD data items and their local values. The counterparty can respond with an acceptance or rejection of the suggested changes.

6.3. Decomposition of the Data Manager component

The Data Manager component provides three basic functionalities. It notifies external parties and the Internal Enactor for data updates. It verifies if a request (internal or external) for update of ECD is correct. When a request for update is approved, it performs an update on the ECD. The Data Update Notifier, Verifier, and Updater components are defined to address these functional requirements (see Fig. 15).

6.3.1. Verifier component

The Verifier component receives requests for update of ECD. Requests can be from the counterparty, the Internal Enactor (via the Enactment Mapper), or the External Enactment Server. When a request for ECD update is received and approved by the Verifier, the Verifier forwards the change that has to be applied to the Updater and the Data Update Notifier. Rejected update requests are stored (for evaluation purposes) and the requester of the update is informed for the rejection. The Verifier sends/receives requests to/from the counterparty for information on the status of certain ECD agreed for monitoring.

6.3.2. Updater component

The Updater component is responsible for accessing and modifying the ECD. On request by the Verifier, it updates the ECD.

6.3.3. Data Update Notifier (DUN) component

The Data Update Notifier receives information from the Verifier for an approved update and sends an information message to the counterparty or/and to the Internal Enactor (via the Enactment Mapper). The DUN is used also for

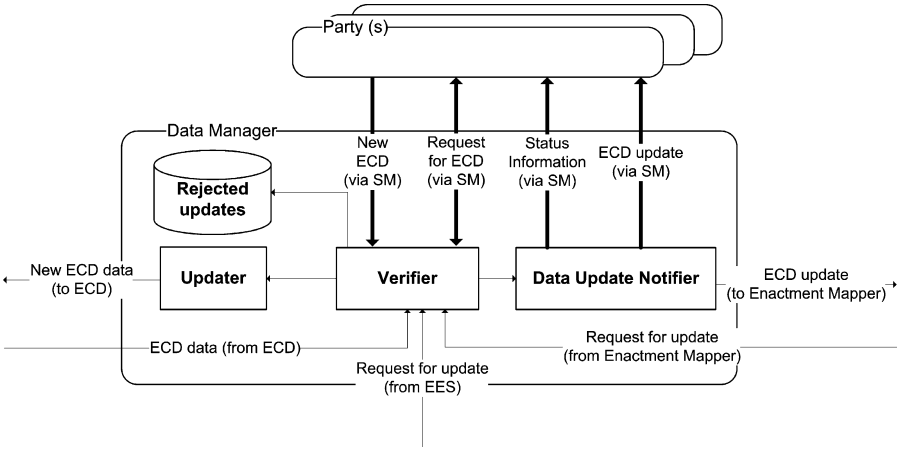

Fig. 15. Decomposition of the Data Manager component.

sending of responses on monitoring requests from the counterparty.

In Sections 4–6, we provided a description of ERA. We started from a high level of abstraction and via functional decomposition we reached a level at which based on existing research and implementation experience components can be further decomposed or even directly implemented. Next, in Section 7, we evaluate ERA.

7. Evaluation of ERA

An evaluation of an architecture helps identifying the strong and weak aspects of the architecture and gives an indication for the success of the system development and implementation processes. A reference architecture serves as a guiding tool for many projects taking place in diverse contexts. Thus, its evaluation prior to its adoption by the stakeholders is of even greater importance. Furthermore, a strong positive evaluation of a reference architecture is an incentive for its wider adoption.

Literature provides a set of methods that can be used to evaluate architectural qualities. Brief summaries and comparison of the most popular methods can be found in Babar and Gorton (2004) and Ionita et al. (2002). To the best of our knowledge, there are no methods dedicated to the holistic evaluation of reference architectures. For the evaluation of the *completeness, modifiability*, *interoperability, integrability, security*, and *flexibility* qualities of ERA, we adapted an existing method and applied it for evaluation of concrete architectures. For the evaluation of the *applicability, usability, feasibility*, and *automation* qualities, we used classical, reasoning techniques.

Next, we provide our results from the evaluation of the completeness, applicability, usability, feasibility, automation, modifiability, interoperability, integrability, security, and flexibility qualities. As we use one method for the evaluation of the modifiability, interoperability, integrability, security, and flexibility qualities, their evaluation is described in one subsection.

7.1. Functional completeness of ERA

In this section, we present our results from the evaluation of ERA in terms of functional completeness of ERA. Use of scenarios is the commonly accepted way for evaluation of completeness of an architecture. For example, the Software Architecture Analysis Method (SAAM) (Bass et al., 2003) is a well-known, scenario-based method to evaluate architectures in terms of completeness and modifiability. Because of the high importance of the functional completeness quality in ERA, we approach the problem in two different, complementary ways, i.e., theoretical and empirical (scenario-based).

7.1.1. Theoretical evaluation of ERA for completeness

We use two conceptual models to evaluate ERA for completeness. The first model describes the contracting activities that an e-contracting system should support. It has already been presented in Section 2.1 and had a leading role in the construction of ERA. Next, we briefly control compliance of ERA with this model.

The main ERA components are defined based on the high-level decomposition of the e-contracting process (see Section 4). In Table 2, we list the e-contracting phases to which an e-contracting process is decomposed in the left column. The ERA components that support these phases are listed in the right column.

In Table 3, we present the mapping between abstract activities (decomposition of the e-contracting phases) and ERA components from the second level of abstraction. As the Matchmaker is not decomposed in ERA, we omit the abstract activities from the information phase.

As it can be seen, ERA provides support for each abstract activity in its design. We stop the analysis of support for e-contracting activities in ERA at this level due to the fact that only few components were decomposed at a third level of detail (i.e., only components whose complexity presents an implementation challenge and no reference architectures exist for their further decomposition).

Next, for evaluation of completeness of ERA, we used a model that defines the fundamental e-contracting concepts. Evaluating the extent to which ERA provides functionalities for the support of the e-contracting concept serves as an additional indication for completeness of ERA. We analyzed ERA from the perspective of the "4W e-contracting framework" (Angelov and Grefen, 2003b), which compared to other contracting frameworks (e.g., Greunz et al., 2000; Griffel et al., 1998) provides higher level of detail and covers a larger set of concepts (Angelov and Grefen, 2003b). To familiarize the reader with the "4W e-contracting framework", we provide a brief summary of it.

Table 2
Support in ERA for the e-contracting phases

Phases	First level components
Information	Matchmaker
Pre-contracting	Partner Selector
Contracting	Contractor
Enactment	Enactor

Table 3
Support in ERA for the e-contracting abstract activities

Abstract activities	Second level components
Partner information	Information collector
Offer	Information collector
Partner selection	Selector
Negotiation	Negotiator
Signing and storing	Contract Finalizer
Value exchange	Internal Enactor
Monitor and control	External Enactment Server and Data Manager
Dispute resolution	Dispute Handler
Evaluation	Evaluator

The central concept in the 4W e-contracting framework is the contract concept. At the highest level of abstraction, the contract concept is associated with four basic concepts, i.e., the *W*ho, *W*here, *W*hat, and Ho*W* concepts. The *W*ho concept models the *actors* that participate in the contract establishment and enactment. The *W*here concept models the *context* of the contract. The *W*hat concept models the *exchanged values* and the *exchange description*. The Ho*W* concept relates to the *means* for contract establishment and enactment. These four general concepts are decomposed as follows:

- The Who concept is decomposed into *party*, *mediator*, and *sub-contractor* concepts.
- The Where concept is decomposed into *business*, *legal*, and *geographical context* concepts.
- The What concept is decomposed into the *product*, *service*, *financial reward*, and *exchange description* concepts.
- The HoW concept is decomposed into the *contracting phase* (information, pre-contracting, contracting, and enactment), *contract representation, contract structure, communication, security*, and *standard* concepts.

Next, we analyze ERA in terms of support for each of the listed e-contracting concepts.

Who. ERA allows contracting between two or more contracting *parties*. Furthermore, ERA allows the involvement of an unlimited number of *mediators* (see Fig. 2) and explicitly defines components that collaborate with a number of mediators (Certificate Authorities, E-notaries, etc.). As *sub-contractors* are a special class of contracting parties, we conclude that ERA allows the involvement of all actors related to e-contracting.

Where. The *legal context* of the contracting relation is defined by the governing law, general provisions, and umbrella contracts. The *business context* is defined by the partner companies and internal business strategies. The legal and business contexts are reflected in ERA by the "Knowledge Updater" and a set of user interface components that allow the maintenance of an up-to-date knowledge with respect to the business and legal environments.

What. The exchanged values (*products, services, financial rewards*) and the *descriptions* for their exchange are defined in the e-contract. The request/offer for a service/product is provided by the Internal Planner, the processes that are supported are provided by the Internal Enactor, and the descriptions for the value exchange are defined via the Definition UI and Design components. The e-contract is established in the Contractor component and is used by the Enactor component. E-contracts are stored in the MC/SA active data component.

HoW. ERA addresses in its design all four *phases of e-contracting*. *Communications* between parties and mediators are depicted through communication channels. The *security* quality on an e-contracting system are addressed by the Secure Messenger and Signer components. As explained in Section 2.2, basic security issues like user authorization and access control are not addressed in ERA. The *standards* used by the parties are provided by different mediators and can be obtained by the Knowledge Updater. The mapping components introduced in ERA provide mappings between different standards. The e-contract is established in a format agreed by the parties during the negotiation process. The Control UI subcomponent of the Contracting Manager can produce human-readable *representation* of e-contracts.

7.1.2. Empirical evaluation of ERA for completeness

As part of the evaluation of ERA, we conducted several meetings with potential users of ERA (see Section 7.6.1 for details). An important part in these meetings was generation of use-case scenarios that ERA should address. The results from these discussions confirmed the need for the functionalities identified in our theoretical approach. Furthermore, stakeholders suggested a number of scenarios which required automation of the maintenance of up-to-date knowledge in the Contracting Manager. Their comments lead us to the definition of the Knowledge Updater component (see Section 5.1). In the initial versions of ERA, the functionalities of the KU were delegated to humans.

In Section 7.2, we discuss the application of ERA for the analysis of existing contracting architectures. Results from this section also demonstrate that ERA addresses the functionalities defined in existing, concrete architectures.

7.2. Applicability of ERA

In this section, we discuss the applicability quality of ERA. We compare the functionalities supported in ERA to the functionalities addressed in existing, domain-independent, commercial and academic architectures. This comparison allows us to demonstrate the applicability of ERA as an analytical tool.

First, we discuss the architectures of two commercial e-contracting systems, i.e., Oracle Contracts and Contracto and compare their functionalities to the functionalities provided by ERA. Oracle Contracts being one of the largest existing contracting applications was chosen for its complexity and size. Contracto was chosen as an example for a small contracting application. Next, we compare the functionalities defined in ERA to the functionalities defined in e-contracting architectures designed in the academia. A number of conceptual e-contracting architectures have been defined in the research world. Many of them incorporate specific technology choices. For example, (Ludwig et al., 2004) concentrates on contracting of Web Services and (Jennings et al., 2000) discusses contracting between software agents for support of workflows. A discussion on the system functionalities required for e-contracting in general settings is presented in the scope of the COSMOS project (Griffel et al., 1998; Merz et al., 1998). However, the publications that resulted from this project did not address the design of an e-contracting architecture. Three

existing conceptual architectures for e-contracting deserve closer attention (i.e., Boulmakoul and Salle, 2002; Hoffner et al., 2001b; Milosevic and Bond, 1995). These architectures aim at automated support of e-contracting in general, technology-independent scenarios. For this reason we compare ERA to these three architectures.

7.2.1. Oracle contracts and ERA

A set of applications for the support of business processes and data management in enterprises is provided in the Oracle E-Business Suite (OES) (Oracle, 2006). The OES contains four dedicated applications for contracting support, i.e., Sales Contracts, Service Contracts, Project Contracts, and Procurement Contracts. An important characteristic of these applications is that they require significant human participation. The differences between them are mostly in the supported, pre-defined contracting scenarios, as well as in the types of contracts the applications can interpret. In this section, we concentrate on the Sales Contracts application and the contracting-related applications integrated with it. The comparison of the other three contracting applications to ERA leads to similar conclusions.

The Sales Contracts application (S) supports the creation of contracts for sales of products. It can receive an offer from the Oracle Quoting application (Q) or an order from the Oracle Order Management application (OM). In both cases, Sales Contracts is used for elaboration of the contract content and its approval by the counterparty. The Oracle Quoting application is used for the elaboration of quotes (offers). It is integrated with the iStore application (iS). The iStore serves as a digital catalogue/shop where the company can publish its products/services.

OES has a couple of applications indirectly related to the Oracle Sales Contracts. The Oracle Marketing application (M) supports marketing of products and services. Oracle Marketing is integrated with iStore. The Oracle Partner Management (PM) application supports maintenance of partners' information. PM can be used by external partners as a match-making engine as well.

The iStore application supports local advertising and matching functionalities of the Matchmaker in ERA. The Oracle Marketing application addresses the external advertising functionality of the Matchmaker. The Oracle Partner Management serves as an advanced external Matchmaker (can be seen as a mediator in ERA). It supports external companies in maintaining their profiles and "matching" them with other companies registered in OPM. Thus, the Matchmaker is implemented by Oracle through the iStore and the Oracle Marketing applications. In addition, the PM application is provided to serve as an external Matchmaker for companies that do not implement locally a Matchmaker. This significant attention on the support of the functionalities of the Matchmaker is due to the practical experience accumulated in the recent years with this type of applications (as discussed in Section 5). The Oracle Sales Contracts application can be mapped to the Contractor component in ERA. Oracle Sales Contracts uses two "supporting" applications, i.e., the Contract Expert Rules and Contract Terms Library, which store information equivalent to the information stored in the "Negotiation rules" and "Management data (C)" sub-components of the Contractor. The Oracle Order Management application provides automation of the management of the order life-cycle. The automation is based on a workflow management system implemented at a lower level of the Oracle architecture. The Oracle Order Management provides support for functionalities of the Contracting Manager in ERA. However, the Oracle Order Management does not cover the complete e-contracting life-cycle. In Table 4, we visualize the mapping of OES contracting components to the components of ERA and the functionalities they support (as defined in Table 1). However, the reader should bear in mind that this visualization is not precise, as often a package addresses functionalities defined in ERA but does not fully support them (e.g., the Order Management package).

From the brief comparison between the contract related applications in OES and ERA, it can be seen that OES does not have complete e-contracting functionalities. OES does not provide functionalities for the evaluation of the best potential partner (part of the Partner Selector in ERA). It does not support management of the complete contracting process but only the process of order creation and contract finalizing. Finally, OES does not support the functionalities of the Enactor component.

7.2.2. Contracto and ERA

The Contracto software (Contracto, 2006) aims at supporting contract management processes in a company. In contrast to OES (see Section 7.2.1), Contracto constitutes of one application. In this application users can have a "global view" on their contractual relationships and a "concrete view" on specific contracts. Contracto allows users to associate contracts with attributes (starting date, status, etc.). Based on these attributes, it can in a limited way monitor the contracting relations and notify users for problems, deadlines, etc. Contracto allows definition

Table 4
Mapping of OES contracting packages to ERA

			OES packages							
			iS	M	PM	Q	OM	S	CER	CTL
ERA components	M	Publish	×	×	×					
		Search	×		×					
	PS	information			×					
		Offer				×	×			
		Selection								
	C	Negotiate						×	×	×
		Sign & store						×		
	E	Exchange								
		Mon.&cont.								
		Dispute								
		Evaluate								
	CM	Manage					×			

Table 5
Mapping of Contracto, BCF, HP, CrossFlow architectures to ERA

			Contracto	BCF	HP	CrossFlow
ERA components	M	Publish				×
		Search				×
	PS	Information	×			
		Offer				×
		Selection				×
	C	Negotiate		×	×	×
		Sign & store		×	×	
	E	Exchange			×	×
		Mon. & cont.		×	×	×
		Dispute Evaluate		×	×	
	CM	Manage	×			×

of contracting activities as well. However, relations between activities cannot be defined. Consequently, the contracting process is managed manually. Contracto allows users to maintain detailed information about former and potential future partners.

Thus, Contracto provides limited support for only two ERA components, i.e., the Contracting Manager and Partner Selector components. However, Contracto provides mainly user interfaces for data representation, while the main functionalities of the Contracting Manager and Partner Selector are performed manually. Table 5 presents a mapping between the functionalities addressed in Contracto and the functionalities delegated to the ERA components. We stress again on the fact that due to the different levels of support for a functionality, this visualization is not precise and is used only to depict the general picture.

As discussed in Section 2.1, contract management is directly linked to each of the four contracting phases. Thus, to support contracting management, Contracto incorporates in its design functionalities required for the support of the e-contracting phases (e.g., selection of partners) and concepts from these phases (e.g., status of the contract negotiation). However, this combination of concepts and functionalities from different e-contracting phases in a single application indicates lack of separation of concerns during system design. Complexity of future improvements and extensions to Contracto will constantly grow as changes will have to be applied on the complete application rather than on specific modules. Contracto does not offer any support for integration with other applications, i.e., interoperability is not addressed in the design of Contracto and is an indication for another weakness in its design.

7.2.3. The Business Contract Framework

The first academic effort for design of an e-contracting architecture (called BCF – Business Contract Framework) is described in Milosevic and Bond (1995). In subsequent publications from this research group, (e.g., Quirchmayr et al., 2002), it is renamed to BCA (Business Contract Architecture). However, the model does not undergo significant changes. BCF can be compared to certain components in ERA. BCF defines a Contract Negotiator and Contract Validator components. These two components can be seen as sub-components of the Negotiator in ERA. The Contract Monitoring and Contract Enforcer components are defined in BCF for the support of contract monitoring and dispute handling. These two components are comparable to the External Enactment Server and Dispute Handler in ERA, respectively. However, in ERA, the EES supports a larger set of functionalities, which are missing in BCF. Mediators like arbitrators, e-notaries, and legal/standard repositories (e.g., contract templates) are addressed in both BCF and ERA. Table 5 shows the mapping between the functionalities addressed in BCF and the functionalities delegated to the ERA components. In conclusion, BCF provides an incomplete representation of the functionalities of an e-contracting system. The main reason for this is the lack at that time of a comprehensive understanding of the e-contracting process and e-contract.

7.2.4. HP e-contracting architecture

The architecture of the e-contracting system presented in Boulmakoul and Salle (2002) is separated into two parts, namely contract establishment and contract enactment parts.

The contract establishment part comprises a Negotiation Engine component, User Interface component and Persistent Store. The Negotiation Engine is equivalent to the Contractor component in ERA. It stores all exchanged contract proposals in the Persistent Store component (equivalent to "Management data (C)").

The contract enactment part comprises the Contract Fulfillment Protocol Manager (CFPM), Reasoner, Scheduler, Contract repository, and Fulfillment components. This part of the architecture can be compared with the decomposition of the Enactor and Secure Messenger components in ERA. The CFPM has similar functionalities as the Communication Monitor component in ERA (see Section 5.5). The Reasoner and Scheduler provide the same functionalities as the External Enactment Server component in ERA (see Section 5.4). In addition, the Reasoner supports communications on contract violations and exchanged enactment information. In ERA, these functionalities are outsourced to the Dispute Handler and Data Manager, respectively. This contributes for higher modifiability of ERA and clearer functionality specification. The Contract repository and the Fulfillment components are equivalent to the MC/SA and Enactment Mapper components, respectively. Thus, the contract enactment part can be seen as similar to the decomposition of the Enactor component in ERA. It misses detailed specification of data management issues (presented in the third level of decomposition of ERA).

The HP architecture does not address management functionalities (provided by the Contracting Manager in ERA). Furthermore, it misses the functionalities supported by the Matchmaker and Partner Selector components in

ERA. Again, Table 5 summarizes our findings on the relation between the functionalities discussed in the architecture of HP and the functionalities defined in ERA.

7.2.5. Crossflow e-contracting architecture

In the CrossFlow project (Hoffner et al., 2001b), a detailed architecture for the support of dynamic outsourcing and contracting is defined. Similar to the HP e-contracting architecture (see Section 7.2.4), the CrossFlow architecture is divided into two main parts, i.e., contract establishment and contract enactment parts.

The first part of the architecture describes the functionalities required for contract establishment. In CrossFlow, the Matchmaker is external for both parties. The Contracting Manager component in CrossFlow implements decision making, negotiation, contract establishment, and contract management functionalities. It can be seen as an aggregation of the functionalities supported by the Partner Selector, Contractor, and part of the Contracting Manager components in ERA. The aggregation of functionalities leads to decreased modifiability of the architecture.

The second part of the CrossFlow architecture supports contract enactment functionalities. The configuration of the enactment components is supported by a dedicated Configuration Manager (in ERA, this functionality is provided by the Enactment Server). It configures: a Proxy-Gateway component that controls the incoming and outgoing messages (implementing the Secure Messenger component in ERA); a Coordinator component that is used to connect the enactment components (similar to the Enactment Server in ERA); Cooperation Support Services (CSS) components that are used to control enactment processes and monitor their quality (similar to the EES in ERA). The modularity of the CSS allows new cooperation services to be easily added to the architecture, which introduces easier modifiability of the enactment architecture. ERA can be extended in this direction through introducing new sub-modules of the EES. In CrossFlow analogs of the Evaluator, Dispute Handler, and Data Manager components are missing. The CSS provide only part of the functionalities of the EES, but are missing its main functionality of steering the contractual relation in the desired direction. The major reasons for this incompleteness are the lack of advanced rule definitions in e-contracts and the lack of explicit attention to the legal aspects of e-contracts and to legal practices (digital signing, dispute resolution, contract updates are not addressed). However, as it can be seen from Table 5, CrossFlow addresses the largest set of functionalities that are defined in ERA.

7.3. Usability of ERA

We presented ERA to both IT and business professionals (see Section 7.6.1 for details). IT experts could easily grasp the major functionalities specified in ERA and the design principles of ERA. In a couple of hours, we managed to explain all three levels of ERA. IT experts found the level of detail satisfactory and could locate specific e-contracting functionalities defined in ERA. After a brief introduction, business professionals could fully comprehend the first level of ERA and the main aspects of the second level (including advanced components like mappers). Our experiments showed that ERA uses naming conventions, notation, and structure that were understandable by the stakeholders. Clearly, as these results were obtained from a limited set of stakeholders, they can be considered only as an indication for the usability quality of ERA.

7.4. Feasibility of ERA

Two fundamental architectural styles are used to achieve *conceptual integrity* in ERA. On the first level of decomposition the layered architectural style provides the overall *structure* of the architecture (see Section 4). Processes in ERA are coordinated by the Management layer. Diverse contracting functionalities are provided by the Application layer. The "Abstract Data Repository" style is used to provide a common way for data exchange between components from different layers (see Sections 5.2–5.5).

Building a prototype will prove its *buildability*. On one hand, the functionalities of certain components of ERA have not yet been implemented in existing software applications. This means that there is no previous experience and knowledge in their development (e.g., design patterns) and algorithms for their operation still have to be defined. Consequently, the easiness of the development of an e-contracting system based on ERA cannot be directly evaluated. On the other hand, functionalities provided by other components of ERA have already been implemented in existing software solutions. The applicability of these existing software solutions for the development of an e-contracting system can be used as a partial indication for the buildability quality of ERA. A summary of the applicability of existing software solutions for the development of an e-contracting system is presented in Angelov and Grefen (2003a). To directly evaluate the buildability quality of the architecture a prototype implementation must be developed. Currently, we investigate the domain of on-line advertising as a potential domain for implementation of a deep e-contracting system based on ERA (Angelov and Grefen, 2006).

7.5. Level of automation of ERA

ERA is aimed at facilitating the design of *highly automated* e-contracting systems. For this reason, in the decomposition of its components, user interfaces are considered only where external knowledge has to be manually introduced to the system (rules, process specifications) or manual handling of exceptions is required. In certain e-contracting systems, a higher degree of human involvement might be preferred. This may be incited by the need to increase flexibility or to avoid costs for automation of activities that can be performed cheaply manually. In these

cases, components can be extended with or even directly replaced by user interfaces that support humans in performing the functionalities of the components. Thus, the level of automation of an e-contracting system has to be evaluated in the concrete design of an e-contracting system. That is why we do not discuss this quality any further.

7.6. Modifiability, integrability, interoperability, security, and flexibility qualities of ERA

The Architecture Tradeoff Analysis Method (ATAM) (Clements et al., 2002) supports evaluation of a wide set of qualities. It is considered to be very suitable for evaluation of modifiability, integrability, and interoperability qualities (Ionita et al., 2002). ATAM concentrates on the relation between the architectural approaches used in an architecture and the desired qualities, and investigates potential sensitive or conflicting points (called "trade-off points"). In this analysis, scenarios (as in SAAM) play a vital role. Based on the analysis, conclusions are drawn on the potentially problematic and positive aspects of the architecture (risks and non-risks).

We used ATAM to evaluate ERA for the modifiability, integrability, interoperability, security, and flexibility qualities. Next, we provide a summary of our results from the evaluation of ERA via ATAM. As ATAM is intended for evaluation of concrete architectures and ERA is a reference architecture, we had to adapt ATAM in a number of points for the evaluation of ERA. We explain the adaptations that we made to ATAM as well.

7.6.1. Following the steps of ATAM

As a first step, we elaborated a list of the architectural approaches (styles, patterns, other reference architectures) used in ERA. The list of approaches for the final version of ERA is provided in Appendix B. Next, we elaborated a utility tree that represents the expected qualities (see Appendix A). The qualities are separated into two classes, i.e. system and architecture qualities (as already explained in Section 2). The qualities that were to be evaluated through ATAM were decomposed to concrete scenarios. Concrete scenarios were ranked for their importance and for complexity of addressing them in ERA.

Next, we organized three small meetings within our research group involving 10 software architects/designers in the first meeting and 3 in the second and third. In the first meeting, we conducted a general, unstructured discussion on ERA. In the second and third meetings, we discussed the qualities, improved the utility tree, and analyzed the suitability of the architectural approaches to the list of qualities. At this point, we found a number of weaknesses in ERA. Each discussion was leading to improvements of ERA and an iteration of the previous steps.

Next, we conducted a meeting with potential business users of ERA. This was one of the problems that we faced in using ATAM. ATAM is used for evaluation of concrete architectures that have a clearly defined group of stakeholders. However, in the case of a reference architecture, there is no concrete group of stakeholders. We identified as main stakeholders a number of roles, i.e., contract managers (this role is given different names like contract engineers, contract supervisors, contract legal officers, etc.), software architects/designers, CIOs, and CEOs. We organized a workshop on e-contracting (Angelov, 2007a), where we presented ERA to 25 representative stakeholders. Though this meeting cannot be representative for all stakeholders of ERA, we consider it sufficient to indicate the level of excellence of ERA (addressing all stakeholders would indeed be impossible). After an adequate introduction, we asked the participants to suggest possible scenarios and prioritize them. We also asked participants to provide their expectations in terms of qualities from ERA. The result was a set of 25 use-case scenarios. We used these scenarios in the evaluation of ERA for completeness as well (see Section 7.1.2). The resulting utility tree was a subset of the qualities identified initially by us. After the workshop, we analyzed the results from it and made final adaptations to ERA.

A general problem that we faced in applying ATAM was the definition of scenarios. Being a reference architecture, ERA is not designed for a specific context, and must address all possible scenarios. Defining concrete scenarios without specific context is difficult and often impossible. That is why we used a "relaxed" format for the scenarios and allowed definition of rather general scenarios.

7.6.2. Main improvements from applying ATAM

The application of ATAM led to a number of improvements of ERA. As already mentioned the Knowledge Updater was introduced as a new component. Furthermore, a number of new architectural styles were introduced, i.e., the Data Indirection style, Abstract Data Repository style, Façade pattern, and the Broker pattern. Finally, we discovered the need of the Internal Mapper component. Thus, ATAM lead to significant improvements in ERA.

7.6.3. Results from applying ATAM

The results from applying ATAM can be divided into two groups, i.e., a structured analysis of the relation between the architectural decisions used in ERA and the required qualities and a set of major conclusions on the strong and weak points of ERA. Next, we summarize the results from applying ATAM.

7.6.3.1. Relation between architectural decisions and qualities. The *modifiability* quality is addressed by the usage of the Layering, Part-Whole decomposition, Abstract Data Repository styles and the Façade, Broker, Batch-sequential, and Mapper patterns.

- The Layering style facilitates achieving higher degree of modifiability. Changes in functionalities influence only "higher" layers. A company may initially implement

few components from the application layer and add additional components in the future. This will have impact only on the Contracting Manager component.
- The Part-Whole decomposition style promotes achieving *separation of concerns* among components, which leads to increased modifiability of a system.
- As explained, in ERA, components from different layers exchange data through databases that implement the Abstract Data Repository style. This style is used to achieve *indirection* in data exchange.
- We expect that a component from the first level of decomposition of ERA (see Fig. 2) will be developed by a single company. Consequently, data integrity between data providers and data consumers becomes of lower importance. For this reason, for data exchange within components, we opted for the Data Indirection style, which also supports high degree of modifiability (easy additions/removal of data consumers/providers) but does not abstract from the format of the stored data. If differences in the data format are to be expected, the Abstract Data Repository can be used.
- The Façade pattern is used to provide *encapsulation* of "lower-layer" components from "higher-layer" components. Using facades allows a "higher-layer" component to be unaware of the concrete decomposition and implementation details of lower-layer components.
- The use of the Broker pattern facilitates achieving *indirection in communications* between internal systems and an ERA-based system. Internal systems can be relocated, added, etc. without affecting ERA but only its Internal Broker.
- The use of the Batch-Sequential pattern allows easy modification to be applied on the Secure Messenger. As this component may be shared with other systems that require external communications, it can be easily adapted to fit their requirements.
- The Mapper pattern is used to define components that "absorb" changes in the external environment with respect to data and process semantics (see the Internal Mapper, External Mapper, and Enactment Mapper). This pattern reduces the amount of modifications required by an ERA-based system when modifications in external data occur.

The Mapper pattern is used to address the *interoperability* quality as well. Mapping components support the usage of different data and process specifications in an e-contracting system and external/internal systems, which leads to increased interoperability.

The *integrability* attribute is addressed by the usage of the Façade pattern and Abstract Data Repository style.

- In the facade of a component the methods and parameters are grouped together and offered for access hiding internal component implementation. An agreement on a façade allows separate development of components and their easier integration.
- The usage of the Abstract Data Repository style allows data producers and consumers to exchange data unaware of each others requirements on the data format.

The *security* quality is addressed by the introduction of the Secure Messenger and the optional Signer components. Manipulation and storage of sensitive business data (contracts and contracting data) is delegated to dedicated components, i.e., Updater and Contract distributor (MC/SA).

The *flexibility* quality is addressed by the introduction of an Enactment Server component that supports the flexible execution of the contracting process. ERA-based systems can establish relationships with diverse parties and can make use of diverse mediators.

7.6.3.2. Sensitivity points, tradeoff points, risks, and non-risks. Sensitivity points and tradeoff points are points in the architecture where the satisfaction of one or more qualities may be influenced by certain properties of the architectural styles or of the components. Risks and non-risks are based on the identified sensitivity and tradeoff points and present a high-level summary of the good and problematic architectural decisions (Clements et al., 2002). Next, we provide a summary of our main findings for the sensitivity/tradeoff points and risks/non-risks in ERA.

We have identified all mapping components as sensitivity points. Mappers can map data and processes between standards that have a common conceptual foundation. It may not be possible to absorb in a mapper external messages/protocols that differ conceptually from the internally used standards. The limitations of mappers can decrease the interoperability quality of ERA. The complexity of the ES, Design, Negotiator, and EES components may hinder development of an ERA-based system (and thus, affect the buildability and automation qualities of ERA).

The tradeoff points are mainly related to the performance of an ERA-based system. Though this quality is not addressed in ERA, we briefly present these tradeoff points (in order to facilitate the development of the other views of an e-contracting system). Security related components (Authenticator, Message Verifier, Cryptographer, and Signer) are computationally intensive components that will be subject to significant loads (Angelov et al., 2005). Similarly, complex mappers will require substantial computational power and may cause decrease in performance. The choice of the Layering style for the structure of ERA affects performance as it requires communication to take place only between neighboring layers. The usage of the Abstract Data Repository style can affect performance as well.

Based on our findings on the sensitivity and tradeoff points, we identified three main risk factors:

- The Secure Messenger is a potential bottleneck for the high performance of an ERA-based system.
- The mapping components may be insufficient to incorporate substantial differences which will lead to changes in internal components.

- The highly advanced functionalities of certain components may be difficult to implement.

Next, we list a number of non-risk factors that we identified in ERA:

- The introduction of the Signer component decreases the load on the Authenticator which leads to improved security management (in a distributed environment) and performance.
- The usage of the Layered style leads to the definition of only three layers. The small amount of layers decreases the negative impact of this style on the performance quality. Furthermore, our meetings showed that it has no negative effect on the usability of ERA by business professionals.
- As indicated in Bass et al. (2003), part-whole decomposition has no negative effect on other qualities.

Up to this point, we have described our main results from the evaluation of ERA. In Section 7.7, we provide a discussion on the evaluation process.

7.7. Discussion

In this section, we analyzed ERA in terms of the qualities expected from an e-contracting reference architecture. Our findings show that ERA addresses all qualities. It excels in the completeness, applicability, and usability qualities. The application of ERA to commercial and academic systems shows that ERA has equivalent components to the components defined in the architectures of each of these systems. However, none of these systems exceeds ERA in terms of completeness and level of detail. Thus, ERA is a contribution to the existing knowledge in the e-contracting domain. Furthermore, the application of ERA to commercial and academic systems shows that ERA can be applied for the analysis of diverse e-contracting architectures described at different levels of abstraction. This illustrates the power of ERA as a tool for analysis of existing e-contracting architectures.

Buildability of ERA (part of its feasibility) is a quality that requires attention during the design of concrete e-contracting systems. Inability of software developers to implement certain advanced and complex functionalities defined in ERA will affect the level of automation of an ERA-based system and will limit the introduction of new business and organizational models offered by e-contracting (Angelov and Grefen, 2004b). Thus, the complexity of the software development process has to be carefully evaluated by the software development company. The performance quality has to be paid specific attention as well. A number of components may affect performance of the system which may limit introduction of business models like just-in-time contracting (Angelov and Grefen, 2004b). The performance quality has to be addressed in a process view of an e-contracting system.

Many designs exist of reference architectures in various application domains of information systems. A few examples are the well-known ISO/OSI network reference model (Tanenbaum, 1992), the Workflow Reference Model (Hollingsworth, 1995), and the Reference Architecture for Workflow Management Systems (Grefen and Remmerts de Vries, 1998). Similar to each of these models, ERA aims at providing a description of the main functionalities that the system must support and a standardized view on the components required for the delivery of these functionalities. Many of the existing reference architectures provide only a high-level view of the system to be designed. These reference architectures serve mostly as standardization models that promote the modular development of a system and its future integration with other information systems. In contrast to them, ERA provides a detailed description of its components. This allows software designers and developers to reach deeper understanding for the components of an e-contracting system and for the interactions among them. As currently no advanced e-contracting systems exist in business practices, this detailed representation of the functionalities of an e-contracting system is a major necessity for both designers and developers.

The process of evaluation of ERA was significantly hindered by the lack of a dedicated method for evaluation of reference architectures. Our experiences during the evaluation of ERA showed that methods for evaluation of concrete architectures (e.g., ATAM) can be applied with limited success for evaluation of a reference architecture. Communications with stakeholders and generation of scenarios play a major role in these methods. However, these steps are hard to perform in the evaluation of reference architectures. Furthermore, these methods do not address the evaluation of architectural qualities that are of fundamental importance in the case of reference architectures. In the evaluation of ERA, we used results from the application of the ATAM method and combined them with our findings obtained by simple reasoning techniques. The combination of all results in a final evaluation was mostly based on our intuition. The existence of a dedicated method for evaluation of reference architectures would significantly substantiate the evaluation process.

8. Conclusions

In this paper, the design of an e-contracting reference architecture (ERA) is presented. The architecture design is guided by well-established, scientific, design principles. ERA provides a detailed description of the system functionalities and qualities that have to be addressed in the implementation of an e-contracting system.

ERA introduces a number of benefits to software developers and business professionals. It allows faster development of e-contracting systems. Furthermore, it introduces standardized view on e-contracting systems (allowing modular system development and facilitating interoperability with other information systems) and will lead to improved

understanding of e-contracting systems. Depending on the people that use ERA and on the goals aimed with its usage, ERA will be used in different ways. It may be expected that the first level of decomposition of ERA (see Section 4) will be most useful for communicating the main architecture principles between software developers and/or business professionals. This level will be used for initial discussions on and general analysis of concrete e-contracting architectures. Lower levels of detail will be mostly valuable for software developers and system integrators who will require a detailed description of the functionalities of an e-contracting system for its design and implementation.

ERA is aimed at facilitating the design of logical views of concrete e-contracting systems. Based on our evaluation of ERA, we believe that it will have a significant contribution to the design process of concrete e-contracting systems and will promote developments in the domain of e-contracting.

Further research work is necessary to investigate the potential problems and solution in the design of process, implementation, and deployment views of e-contracting systems as suggested in Kruchten (1995). For example, flexibility of contracting processes and multiple, simultaneously running contracting processes are factors that require specific attention in the design of a process view of an e-contracting system.

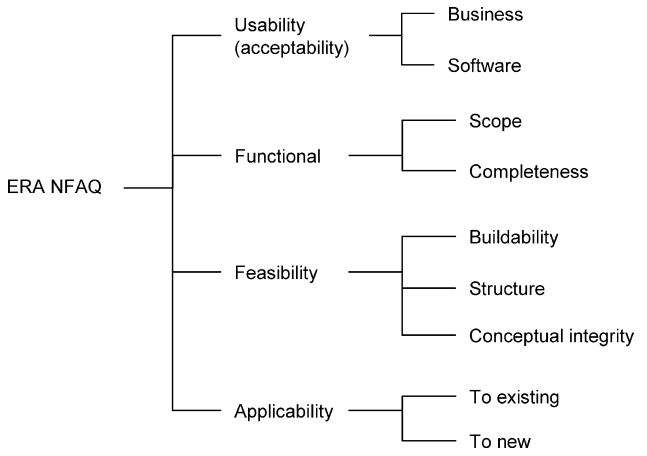

Fig. 17. Utility tree for non-functional architectural qualities (NFAQ).

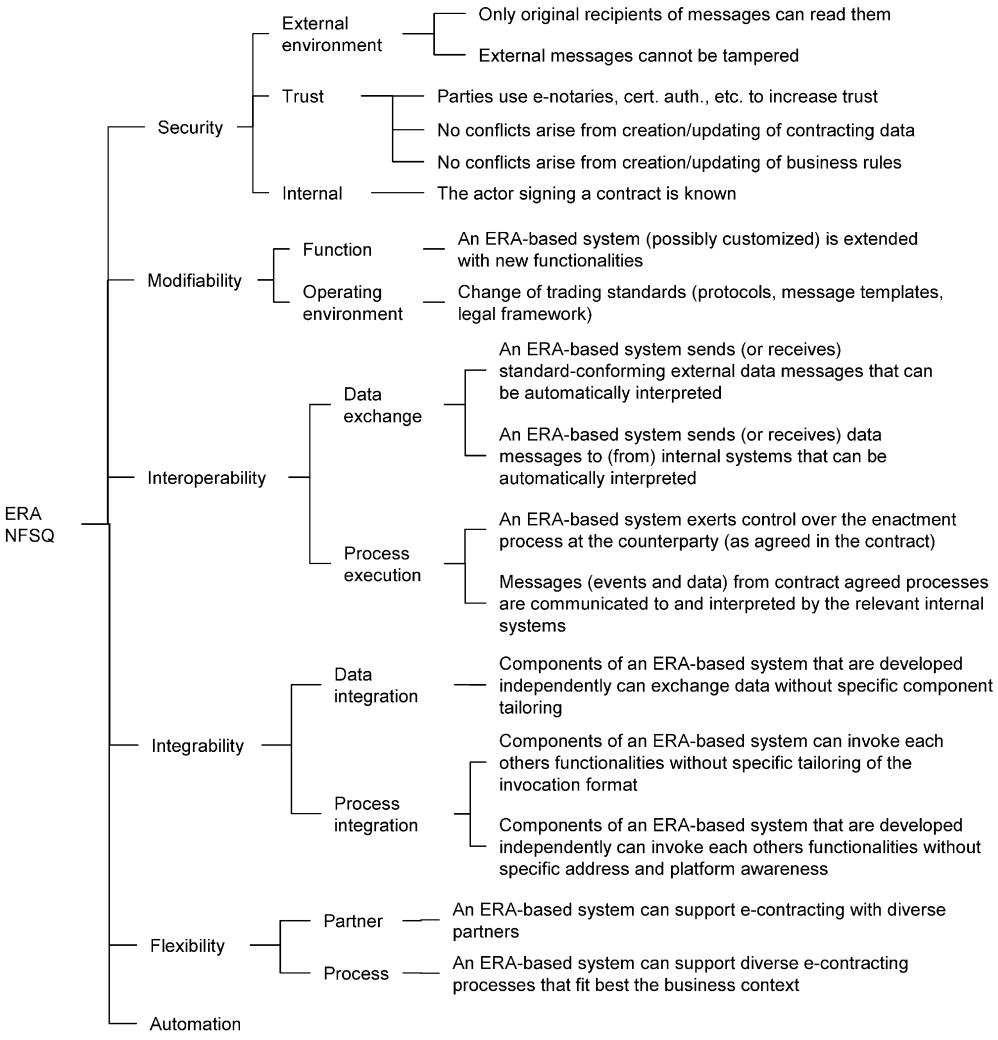

Fig. 16. Utility tree for non-functional system qualities (NFSQ).

Appendix A. Utility trees for non-functional qualities

Next, we list the utility trees that represent the required qualities in ERA. In Fig. 16, we list the non-functional system qualities. As qualities from this part of the utility tree are evaluated by the Architecture Tradeoff Analysis Method (ATAM), the utility tree is decomposed to a level of concrete scenarios.

Fig. 17 presents the utility tree for non-functional architectural qualities.

Appendix B. List of architectural approaches

B.1. Architectural approaches at the first level of ERA

- Layering style (Bass et al., 2003; Klein and Kazman, 1999).
- Part-whole decomposition style (Bass et al., 2003).
- Facade pattern (Gamma et al., 1995).

B.2. Architectural approaches at the second level of ERA

- WFMC standard architecture (Hollingsworth, 1995) and RAWFMS (Grefen and Remmerts de Vries, 1998). Related components: *Contracting Manager*.
- Batch sequential pattern (Bass et al., 2003). Related components: *Secure Messenger*.
- Data indirection style (Klein and Kazman, 1999). Related components: *Selection rules, Negotiation rules, Enactment rules, Management data, Application data, Process data, Process, rule, and ontology definitions*.
- Abstract data repository style (Klein and Kazman, 1999). Related components: *Invalid messages, Management data (PS), Evaluations, Management data (C), MC/SA, Management data (E), Management data (SM), Rejected updates*.
- Broker pattern (Buschmann et al., 1996). Related components: *Internal broker*.
- Mapper pattern (Fowler, 2002). Related components: *External Mapper, Internal Mapper, Enactment Mapper*.
- Security dedicated components – *Cryptographer, Authenticator, Message Verifier, Signer, Communication Monitor, Updater, Contract Distributor (MC/SA)*.

References

Angelov, S., 2006. Foundations of B2B Electronic Contracting. Technische Universiteit Eindhoven.

Angelov, S., 2007a. Defining e-contracting and measuring its significance. In: Post-conference Workshop at the 4th Annual Contract Management Conference. Institute for International Research, Dubai.

Angelov, S., 2007b. Evaluation of the E-contracting Reference Architecture, Beta Working Paper, Eindhoven University of Technology.

Angelov, S., Grefen, P., 2001. B2B eContract handling – a survey of projects, papers and standards, CTIT Technical Reports, University of Twente.

Angelov, S., Grefen, P., 2002. Support for B2B e-contracting – the process perspective. In: Marik, V., Camarinha-Matos, L.M., Afsarmanesh, H. (Eds.), Knowledge and Technology Integration in Production and Services: Balancing Knowledge in Product and Service Life Cycle, Fifth IFIP/IEEE International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS'02), vol. 229, September 25–27, 2002, Cancun, Mexico, Kluwer, pp. 87–96.

Angelov, S., Grefen, P., 2003a. An analysis of the B2B E-contracting domain: paradigms and required technology, Beta Working Paper, WP 102, Eindhoven University of Technology, Eindhoven.

Angelov, S., Grefen, P., 2003b. The 4W framework for B2B e-contracting. International Journal of Networking and Virtual Organisations 2, 78–97.

Angelov, S., Grefen, P., 2004a. Supporting the Diversity of B2B E-contracting Processes, Telematica Instituut Technical reports, TI/RS/2003/120, Telematica Instituut.

Angelov, S., Grefen, P., 2004b. The business case for B2B e-contracting. In: Janssen, M., Sol, H.G., Wagenaar, R.W. (Eds.), Proceedings of the 6th International Conference on Electronic Commerce, Delft, The Netherlands, vol. 60. ACM Press, New York, pp. 31–40, October 25–27.

Angelov, S., Grefen, P., 2005. Requirements on a B2B E-contract Language, Beta Working Paper, WP 140, Eindhoven University of Technology.

Angelov, S., Grefen, P., 2006. A case study on electronic contracting in on-line advertising – status and prospects. In: Camarinha-Matos, L., Afsarmanesh, H., Ollus, M. (Eds.) Network-Centric Collaboration and Supporting Frameworks – Proceedings 7th IFIP Working Conference on Virtual Enterprises, 25–27 September 2006, Helsinki, Finland, Springer, Boston, pp. 419–428.

Angelov, S., Till, S., Grefen, P., 2005. Dynamic and secure B2B e-contract update management. In: Kearns, M., Reiter, M. (Eds.), Proceedings of the 6th ACM Conference on Electronic Commerce. ACM Press, New York, Vancouver, BC, Canada, pp. 19–28, June 5–8.

Babar, M., Gorton, I., 2004. Comparison of scenario-based software architecture evaluation methods. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04). IEEE Computer Society, Washington, USA, pp. 600–607.

Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice, second ed. Addison-Wesley Professional.

Beam, C., Segev, A., 1997. Automated negotiations: a survey of the state of the art. Wirtschaftsinformatik 39, 263–268.

Bichler, M., Segev, A., 1999. A brokerage framework for Internet commerce. Distributed and Parallel Databases 7, 133–148.

Booh, G., Rumbaugh, J., Jacobson, I., 1999. The Unified Modeling Language User Guide. Addison-Wesley Longman.

Boulmakoul, A., Salle, M., 2002, Integrated Contract Management, HP Labs Technical Reports, HPL-2002-183, HP Laboratories Bristol.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., 1996. Pattern-Oriented Software Architecture, Volume 1: A System of Patterns. John Wiley & Sons.

Chiu, D.K.W., Cheung, S.C., Till, S., 2003, A three-layer architecture for e-contract enforcement in an e-service environment. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) – Track 3, vol. 3, Big Island, Hawaii IEEE Computer Society, Washington, pp. 74.

Clements, P., Kazman, R., Klein, M., 2002. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley Professional.

Contracto. 2006. Contracto software, version 6.02, <http://www.contracto.nl/>.

Dan, A., Dias, D., Nguyen, T., Sachs, M., Shaikh, H., King, R., Duri, S., 1998. The Coyote project: framework for multi-party e-commerce. Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries, Heraklion, Crete, Greece, vol. 1513. Springer-Verlag, London, pp. 873–889.

Dhillon, G., 2007. Principles of Information Systems Security: Text & Cases. Wiley.

Fowler, M., 2002. Patterns of Enterprise Application Architecture. Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Grefen, P., Aberer, K., Hoffner, Y., Ludwig, H., 2000. CrossFlow: cross-organizational workflow management in dynamic virtual enterprises. International Journal of Computer Systems Science and Engineering 15, 277–290.

Grefen, P., Remmerts de Vries, R., 1998. A reference architecture for workflow management systems. Data & Knowledge Engineering 27, 31–57.

Grefen, P., Ludwig, H., Angelov, S., 2003. A three-level framework for process and data management of complex e-services. International Journal of Cooperative Information Systems 12, 487–531.

Greunz, M., Schopp, B., Stanoevska-Slabeva, K., 2000. Supporting market transactions through XML contracting container. In: Proceedings of the Sixth Americas Conference on Information Systems (AMCISS 2000), August 10–13, Long Beach, CA.

Griffel, F., Boger, M., Weinreich, H., Lamersdorf, W., Merz, M., 1998. Electronic contracting with COSMOS – how to establish, negotiate and execute electronic contracts on the Internet. In: Proceedings of the 2nd International Workshop on Enterprise Distributed Object Computing (EDOC'98), November 3–5, San Diego, pp. 46–55.

Hoffner, Y., Field, S., Grefen, P., Ludwig, H., 2001a. Contract driven creation and operation of virtual enterprises. Computer Networks – the Int. Journal of Computer and Telecommunications Networking 37, 111–136.

Hoffner, Y., Ludwig, H., Grefen, P., Aberer, K., 2001b. CrossFlow: integrating workflow management and electronic commerce. ACM SIGecom Exchanges 2, 1–10.

Hollingsworth, D., 1995. The Workflow Reference Model, Workflow Management Coalition Documents, TC00-1003, Workflow Management Coalition.

ILOG, 2006. ILOG JRules, <http://www.ilog.com/products/jrules/>.

Immonen, A., Niemelä, E., Matinlassi, M., 2005. Evaluating the integrability of COTS components – software product family viewpoint. In: Beydeda, S., Gruhn, V. (Eds.), Testing Commercial-off-the-Shelf Components and Systems. Springer, Berlin, Heidelberg, pp. 141–167.

International Organization for Standardization, 2006. ISO/IEC FCD 9126-1: Information Technology – software quality characteristics and metrics – Part 1: Quality characteristics and subcharacteristics.

Ionita, M., Hammer, D., Obbink, H., 2002. Scenario-based software architecture evaluation methods: an overview. In: Workshop on Methods and Techniques for Software Architecture Review and Assessment at the International Conference on Software Engineering, Orlando, FL, USA.

Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Odgers, B., 2000. Autonomous agents for business process management. International Journal of Applied Artificial Intelligence 14, 145–189.

Klein, M., Kazman, R., 1999. Attribute-based architectural styles, Carnegie Mellon University Technical Reports, CMU/SEI-99-TR-022, Carnegie Mellon University.

Kruchten, P., 1995. Architectural blueprints – the "4+1" view model of software architecture. IEEE Software 12, 42–50.

Ludwig, H., Dan, A., Kearney, R., 2004. Cremona: An architecture and library for creation and monitoring of WS-agreements. In: Proceedings of the 2nd International Conference on Service Oriented Computing. ACM Press, New York, pp. 65–74.

Maciaszek, L.A., 2001. Requirements analysis and system design. Developing Information Systems with UML. Addison Wesley.

Masum, H., Zhang, Y., 2004. Manifesto for the reputation society. First Monday 9.

Merz, M., Griffel, F., Tu, T., Müller-Wilken, S., Weinreich, H., Boger, M., Lamersdorf, W., 1998. Supporting electronic commerce transactions with contracting services. International Journal on Cooperative Information Systems 7, 249–274.

Milosevic, Z., Bond, A., 1995. Electronic commerce on the Internet: what is still missing? In: Proceedings of the 5th Annual Conference of the Internet Society, INET'95, Honolulu, Hawaii.

Milosevic, Z., Jøsang, A., Dimitrakos, T., Patton, M.A., 2002, Discretionary enforcement of electronic contracts. In: Proceedings of the Sixth International Enterprise Distributed Object Computing Conference, September 17–20, 2002, Lausanne, Switzerland, IEEE Computer Society, pp. 39–50.

Nahmias, S., 1997. Production and Operations Analysis. McGraw-Hill.

Oracle, 2006. Oracle E-Business Suite, Release 11.5.10.2. <http://www.oracle.com/applications/e-business-suite.html>.

Papazoglou, M., 2007. Web Services: Principles and Technology. Prentice Hall.

Petritsch, H., 2006. Service-Oriented Architecture (SOA) vs. Component Based Architecture. Vienna University of Technology, Vienna.

Quirchmayr, G., Milosevic, Z., Tagg, R., Cole, J., Kulkarni, S., 2002. Establishment of virtual enterprise contracts. In: Hameurlain, A., Cicchetti, R., Traunmuller, R. (Eds.), Vienna University of Technology, vol. 2453. Springer-Verlag, London, pp. 236–248.

Rozanski, N., Woods, E., 2005. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison-Wesley Professional.

Savvion, 2006, Savvion Business Rules. <http://www.savvion.com>.

Tanenbaum, A.S., 1992. Modern Operating Systems. Prentice Hall.

Turban, E., Lee, J., King, D., Chung, H., 2000. Electronic Commerce – A Managerial Perspective. Prentice Hall.

Wigand, R., Picot, A., Reichwald, R., 1997. Information, Organization and Management: Expanding Markets and Corporate Boundaries. John Wiley and Sons Ltd.

Zdravkovic, J., Kabilan, V., 2005. Enabling business process interoperability using contract workflow models. In: Meersman, R., Tari, Z., Hacid, M.-S., Mylopoulos, J., Pernici, B., Babaoglu, O., Jacobsen, H.-A., Loyall, J.P., Kifer, M., Spaccapietra, S. (Eds.), On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, vol. 3761, October 31–November 4, 2005, Agia Napa, Cyprus, Springer, pp. 77–93.

Samuil Angelov is an assistant professor in the Department of Technology Management at Eindhoven University of Technology. In the period 1998–2003, he worked for the Bulgarian Academy of Sciences, Sofia University, and Twente University. He received his Ph.D., in February 2006 at the Technical University of Eindhoven. During and shortly after his master studies, he was involved in the European INCO COPERNICUS project "MALL 2000". During his Ph.D. studies, he participated in the SOBI project managed by the Telematica Institute. In his Ph.D. studies, he concentrated on the design of an information system for highly automated electronic contracting. His research work led to a number of international publications. He is regularly involved in reviews of papers for international conferences and journals. He is a member of the international association SOCOLNET. His research interests are in the field of enterprise and information system modeling and concentrate on the domain of electronic commerce and electronic contracting.

Paul Grefen is a full professor in the Department of Technology Management at Eindhoven University of Technology, where he chairs the Information Systems subdepartment and leads the ICT Architectures group. He received his Ph.D., in 1992 from the University of Twente. From 1992 until early 2003, he held assistant and associate professor positions in the Computer Science Department at the University of Twente. He was a visiting researcher at Stanford University in 1994. He was involved in the WIDE ESPRIT project, which focused on advanced database support for workflow management systems, and the CrossFlow IST project, which aimed at cross-organizational workflow support for dynamic virtual enterprises. Currently, he is involved in the CrossWork IST project focusing on process support in the automotive industry and the XTraConServe NWO project working on contracted transactional services. He has been a member of the program committees of a large number of international conferences and a regular reviewer for several international journals. He is an associate editor of the International Journal of Cooperative Information Systems. His current research interests include architectural design of complex information systems, high-level transaction management, advanced workflow management, and contract support in electronic business.