COMPUTING SCIENCE

A Declarative Approach to Configuring Business-to-Business Conversations

Derek Mortimer and Nick Cook

TECHNICAL REPORT SERIES

No. CS-TR-1214 **July 2010**

TECHNICAL REPORT SERIES

No. CS-TR-1214 July, 2010

A Declarative Approach to Configuring Business-to-Business Conversations

D. Mortimer, N. Cook

Abstract

Business-to-business (B2B) interactions typically involve the exchange of documents and messages to achieve specific goals. This has led to the development of standards allowing the specification of complete electronic conversations detailing the syntax, semantics and sequencing of messages to be exchanged. As with paper based interactions, electronic conversations may be subject to agreements between partners, implying that both individual messages and entire conversations may have specific requirements imposed upon them. Ideally, an enterprise should be able to specify these requirements in a manner that is both declarative, freeing them from underlying technical concerns to focus solely on meeting their business obligations, and compatible with existing B2B conversation standards, to minimise re-engineering of existing business processes. A common such example is accountability, where all participants must be held accountable for their actions during an exchange. Accountability can be satisfied through the use of specific exchange protocols and digitally signed evidence. However, business level clients may not, and should not, have to be aware of these technical concerns when they can be automated at lower levels of operation. This paper describes an approach to transparently satisfying requirements for B2B conversations conducted using message oriented middleware, providing accountability support as a major example.

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

MORTIMER, D., COOK, N.

A Declarative Approach to Configuring Business-to-Business Conversations
[By] D. Mortimer, N. Cook
Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2010.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-1214)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1214

Abstract

Business-to-business (B2B) interactions typically involve the exchange of documents and messages to achieve specific goals. This has led to the development of standards allowing the specification of complete electronic conversations detailing the syntax, semantics and sequencing of messages to be exchanged. As with paper based interactions, electronic conversations may be subject to agreements between partners, implying that both individual messages and entire conversations may have specific requirements imposed upon them. Ideally, an enterprise should be able to specify these requirements in a manner that is both declarative, freeing them from underlying technical concerns to focus solely on meeting their business obligations, and compatible with existing B2B conversation standards, to minimise re-engineering of existing business processes. A common such example is accountability, where all participants must be held accountable for their actions during an exchange. Accountability can be satisfied through the use of specific exchange protocols and digitally signed evidence. However, business level clients may not, and should not, have to be aware of these technical concerns when they can be automated at lower levels of operation. This paper describes an approach to transparently satisfying requirements for B2B conversations conducted using message oriented middleware, providing accountability support as a major example.

About the authors

Derek Mortimer is currently undertaking a PhD involving supporting business to business interactions through the use of middleware services, more recently in Service oriented and Cloud computing contexts. The PhD is funded by an EPSRC and Industry scholarship with Red Hat.

Dr. Nick Cook is a Lecturer and member of the School's Systems Research Group. His research interests concern enterprise and inter-enterprise middleware, focussing on practical support for non-repudiable interactions between organisations. Developing novel middleware for both non-repudiable information sharing and message delivery.

Suggested keywords

B2B EXCHANGE
NONREPUDIATION
ACCOUNTABILITY

A Declarative Approach to Configuring Business-to-Business Conversations

Derek Mortimer and Nick Cook
{d.j.mortimer,nick.cook}@ncl.ac.uk

University of Newcastle-upon-Tyne

Abstract. Business-to-business (B2B) interactions typically involve the exchange of documents and messages to achieve specific goals. This has led to the development of standards allowing the specification of complete electronic conversations detailing the syntax, semantics and sequencing of messages to be exchanged. As with paper based interactions, electronic conversations may be subject to agreements between partners, implying that both individual messages and entire conversations may have specific requirements imposed upon them. Ideally, an enterprise should be able to specify these requirements in a manner that is both declarative, freeing them from underlying technical concerns to focus solely on meeting their business obligations, and compatible with existing B2B conversation standards, to minimise re-engineering of existing business processes. A common such example is accountability, where all participants must be held accountable for their actions during an exchange. Accountability can be satisfied through the use of specific exchange protocols and digitally signed evidence. However, business level clients may not, and should not, have to be aware of these technical concerns when they can be automated at lower levels of operation. This paper describes an approach to transparently satisfying requirements for B2B conversations conducted using message oriented middleware, providing accountability support as a major example.

1 Introduction

It is increasingly common to structure business-to-business (B2B) interactions in terms of well-defined business message exchanges between loosely coupled services. To this end open standards such as RosettaNet [14] and ebXML [11] have been developed. These standards allow observable B2B conversations to be defined in terms of the messages that must be exchanged in order to achieve business objectives. Partners use such standards as the basis for agreement on syntax, semantics and sequencing of messages; and the business processes upon which they should execute.

This message based interaction presents message oriented middleware as the ideal approach, providing a loosely coupled and asynchronous solution to message delivery while also allowing additional functionality, declarative business level requirements for example, to be delivered transparently to the end users.

As with their paper based analogues, electronic B2B interactions may be subject to agreements between participants, implying that both individial messages and entire conversations may have specific requirements imposed upon them. At the business level, these requirements should be both declarative, allowing participants to free themselves from underlying technical concerns and focus entirely on meeting their business obligations, and compatible with existing standards to promote transparent interoperation and minimise the re-engineering of existing business processes.

Taking *accountability* as major example requirement, participants may require that all collaborators in an interaction be held to account for their actions. We can choose to satisfy accountability through the use of specific fair exchange protocols involving digitally signed *non-repudiation* evidence. Non-repudiation being formally defined as the inability for any participant to subsequently falsely deny their participation in a specific exchange [5].

It is neither desirable, nor required, that business clients are aware of *how* their requirements are being satisfied, they simply require that they are being satisfied. We say it is neither desirable nor required as it is possible to derive technical level requirements through the transformation of business level requirements using information including configuration, and available knowledge and mappings.

It follows though, that for each declarative requirement to be supported, necessary primitives must exist at lower levels of operation to facilitate their satisfaction, for example, accountability may require non-repudiation and fair exchange protocols and confidentiality may require specific encryption and signature algorithms.

This paper provides a technique for transparently supporting declarative business level requirements on exchanges carried out over message oriented middleware using an extensible message gateway. Accountability will be supported as an example requirement, satisfied through the implementation of fair non-repudiation protocols and reliable evidence logging.

Section 2 provides a more thorough background to the problems tackled in this paper and the techniques employed. Section 3 describes the design briefly discusses the implementation. Section 4 describes related work and Section 5 presents the results, conclusions and future work.

2 Background

2.1 Business-to-Business Exchange Standards

Standardisation such as RosettaNet Partner Interface Process and Implementation Framework [14,13] and Electronic Business XML (ebXML) [11,10] has allowed the specification of well defined B2B conversations at higher levels than previously possible, we refer to these as conversations or exchanges interchangeably.

These standards support the specification of requirements on entire conversations and individual messages at varying levels of flexibility. RosettaNet, for

example, accomodates binary attributes specifying non-repudiation or reliable delivery and mandates the specific technical requirements to satisfy them[1] but makes no provision for supporting additional arbitrary meta-data regarding delivery. ebXML on the other hand supports both these binary style attributes but also allows messages to be extensible and include arbitrary data, providing it does not alter the observable business semantics.

While the situation is perhaps not ideal in cases such as RosettaNet, it provides a useful lowest common denominator in terms of supporting additional requirements in a manner compatible with existing standards, specifically when using message oriented middleware where intermediares are permitted.

2.2 Message oriented Middleware

There are various middleware specifications facilitating the execution of interactions through the exchange of self-contained messages, we refer to these approaches as message oriented middleware. This approach has inherent advantages such as asynchronicity and loose coupling and allows the routing of messages through intermediaries to provide extended functionality in a manner transparent to both the sender and recipient.

To participate in a B2B conversation here a party simply has to construct a valid business message, as per some well defined conversation, and pass it to the middleware for delivery. The asynchronous nature of the system means recipients need not be online at the time of transmission of a message and senders need not necessarily remain online after the message has been sent, this is particularly useful for both the long-lived nature of some B2B conversations and allowing these interactions to be fault tolerant.

Most message oriented middleware supports the routing of messages through intermediaries en route to their destination, this ability to intercept messages is the enabling technique that allows additional functionality to be transparently delivered and enables the realisation of the design outlined in this paper.

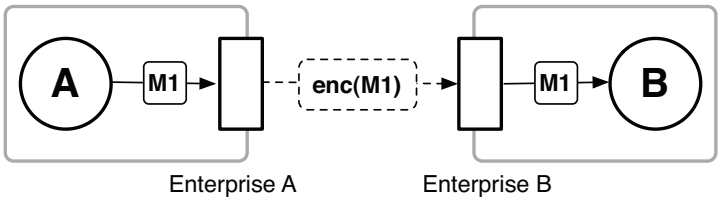

Fig. 1. Message M_1 is intercepted at the boundaries of *Business A* and *Business B* to ensure the contents are encrypted whilst travelling between them.

[1] E.g., RosettaNet Implementation Framework specifies that Voluntary Non-Repudiation, as discussed later, must be used for the evidence exchange.

Figure 1 illustrates a simple but very useful example of interception providing transparent additional functionality, the message is intercepted before it leaves *Business A* and encrypted, either by transforming M_1 directly or encapsulating it in a container message. The message remains encrypted until it enters *Business B* at which point it is automatically decrypted and delivered as a plain message. Neither A or B are required to be aware this encryption occured (i.e. transparency is maintained), yet there is assurance, to those who require it, that encryption occured on the wire while all business logic remains completely unchanged.

This ability to transform, restore and encapsulate messages along the wire allows compatability to be maintained with rigid specifications such as RosettaNet while additional requirements can still be satisfied.

2.3 Accountability as a Business Requirement

Accountability is commonly identified as a vital requirement of B2B interactions [3,8,6], where all participants must be held to account for, and in acknowledgement of their actions during an exchange.

Accountability is of particular importance to B2B interactions where collaborating participants are working towards one or more mutually beneficial business goals. Each participant requires that their own interests are protected in the context of the collaboration, specifically that their own contributions are recognized; and that partner collaborators are held accountable for their actions.

Accountability as a declarative requirement can be satisfied through the use of specific non-repudiation and fair exchange protocols alongside the maintenance of an irrefutible evidence log for use in dispute resolution, .

Non-repudiation protocols are responsible for the generation and structured exchange of digitally signed evidence, minimally requiring trusted evidence proving irrefutably that a given message originated from a specific sender and counterpart evidence proving irrefutably that the message was received by the intended recipient.

While non-repudiation is required, it is not sufficient for accountability in B2B interactions. The *selective receipt problem* [2] illustrates an example in which a recipient may refuse to deliver their proof of receipt after they have obtained both a confidential message and its proof of origin evidence, placing the sender at a disadvantage as a direct result of their compliance.

Fairness is formally described by Markowitch et al. [6] as the property that "For a fixed level of quality over the communication channels, at the end of an exchange protocol run, either all involved parties obtain their expected items or none, even a part, of the information to be exchanged is revealed.", it is one of three mandatory properties for any exchange system wishing to classify itself as *secure*.

The incorporation of fairness into non-repudiation protocols ensures the digital evidence is exchanged in a manner that prevents any participant being placed at a disadvantage as a result of correct behaviour, alleviating issues such as the selective receipt problem.

Using the aforementioned to guarantee accountability in a fair manner we can benefit B2B interactions by compelling participants to behave correctly, safe in the knowledge that they will not be disadvantaged as a result of compliance [2,1,16].

2.4 Non-repudiation Classifications

Kremer et al [5] conducted an intensive survey of available non-repudiation protocols and classified each according to level of fairness provided and level of involvement from the TTP. We assume in this paper that non-repudiation is applied for a message exchange between two parties and that no party will act against their own interests. All of the protocols discussed rely on two key types of evidence:

Non-repudiation of Origin (NRO) Evidence that can be used to unambiguously prove that a signatory is the author of a specific message.

Non-repudiation of Receipt (NRR) Evidence that can be used to unambiguously prove that a signatory received a specific message, also containing the corresponding NRO evidence for that message.

The survey goes on to provide a fine grained definition of fairness covering three major classifications. *Weak fairness* ensures that if any party does not receive information or evidence they are entitled to, they at least receive proof of this fact for use in dispute resolution. *Strong fairness* states that all participants are able to retrieve the information and evidence they are entitled to at the end of a protocol run, or none can. *True fairness* includes strong fairness with the added caveat that, for a successful protocol execution, all evidence generated is done so independently of how the protocol executed (i.e. evidence generated without TTP intervention is indistinguishable from when the TTP was invoked to enforce the exchange).

TTP involvement is classified as one of three possibilities including *inline* where the TTP is involved in every message transmission, *online* where the TTP is involved in at least every protocol execution and *offline* where the TTP is involved only when a dispute is raised.

A specialised type of TTP known as a Timestamping Authority (TSA) may be used to provide *trusted timestamps*, proof that a piece of information was witnessed at a specific time by a mutually trusted party. These trusted timestamps are used to ensure that evidence which may have been logged for a long period of time was valid at time of generation, even in the face of revoked certificates and keys, for example.

2.5 Coffey-Saidha Non-repudiation Protocol

The most desirable protocol as per the aforementioned classifications is one providing true fairness through the use of an offline TTP. Such a scheme incurs

significant complexity however and this paper will focus initially on an implementation of the Coffey-Saidha non-repudiation protocol with support for additional protocols to be added seamlessly in the future. The Coffey-Saidha protocol provides strong fairness through the use of an inline TTP and is described in figure 2.

$$
\begin{array}{llcll}
1 & A & \rightarrow & TTP & : & enc_{TTP}\,(id,\, A,\, B,\, m,\, NRO,\, ts_{TSA}\,(NRO)) \\
2 & TTP & \rightarrow & B & : & id,\, A,\, B,\, h\,(NRO) \\
3 & B & \rightarrow & TTP & : & enc_{TTP}\,(id,\, A,\, B,\, NRR,\, ts_{TSA}\,(NRR)) \\
4 & TTP & \rightarrow & B & : & id,\, A,\, B,\, m,\, NRO,\, ts_{TSA}\,(NRO) \\
5 & TTP & \rightarrow & A & : & id,\, A,\, B,\, NRR,\, ts_{TSA}\,(NRR)
\end{array}
$$

Fig. 2. Coffey-Saidha in-line TTP fair exchange protocol.

Table 1 details the notation and definitions required to describe the exchange protocol.

Notation	**Description**
id	unique protocol run identifier
$h\,(x)$	secure hash of x
$i,\, j$	concatenation of items i and j
$P \rightarrow Q\ :\ m$	P sends m to Q
$sig_P\,(x)$	P's digital signature on x
$enc_P\,(x)$	encryption of x with P's public key
T_g	Timestamp for generation of some information
$ts_{TSA}\,(x)$	Timestamp on x generated by a TSA bearing witness to the time of generation of information x

Token	**Definition**
NRO	$sig_A\,(id, A, B, m)$
NRR	$sig_B\,(id, A, B, h\,(NRO))$
$ts_{TSA}\,(x)$	$\{T_g, sig_{TSA}\,(x, T_g)\}$

Table 1. Notation and Definitions for protocol elements.

A successful execution of the Coffey-Saidha protocol using an inline TTP will complete in 5 steps with both parties obtaining both NRO and NRR evidence for the message m being exchanged, the use of encryption in steps 1 and 3 prevents A and B from obtaining NRO and NRR evidence prematurely and gaining an unfair advantage.

A detailed explanation of the protocol and all of its steps, including abort and timeout sub-protocols can be found in [1].

3 Design and Implementation

The design comprises a messaging service (MS) located at the boundary of an organisation. This service acts as a gateway through which all messages are routed and requirements are satisfied, collaborating with other services as necessary, as represented in figure 3.

This *messaging gateway* pattern is well recognised in enterprise system design [4] and is intended to allow additional functionality to be delivered transparently as an intermediary. Non-repudiation will be used to provide accountability as an example of business level requirement support within the gateway.

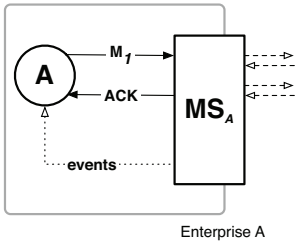

Fig. 3. MS_{A} collaborates on behalf of A with external parties to fairly exchange M_1 and ACK.

Figure 3 introduces an important distinction in the form of *business messages* (illustrated by solid lines) and *protocol* (or *technical) messages* (dashed lines). Clients such as A are only expected to produce and consume business messages and protocol messages are generated by the messaging service. To maintain transparency, clients are never required to understand protocol messages although they may choose to do so.

3.1 Message Processors and Handlers

Within an instance of the messaging service as shown in figure 3, messages that have been intercepted must be processed in order to both assess which requirements have been specified and go about satisfying them. The design does this in terms of *message processors*, intended to be implemented in a modular fashion so they may be logically grouped to render specific functionality.

We refer to this grouping of a set of message processors as a *handler*, each handler is configured with an ordered list of message processors through which messages can be sequentially passed. In order to control flow of messages through the messaging service, handlers are connected via locations such as queues and topics. The design specifies four types of connected locations for a single handler:

Input The mandatory input location specifies where messages that should be intercepted are deposited.

Output The optional output location specifies where messages should be deposited once processing has completed.

Signal The optional signal topic specifies a location into which significant events during a handler's execution are deposited for interested subscribers.

Audit The optional audit topic specifies a location into which verbatim copies of messages generated during a handler's execution are also deposited.

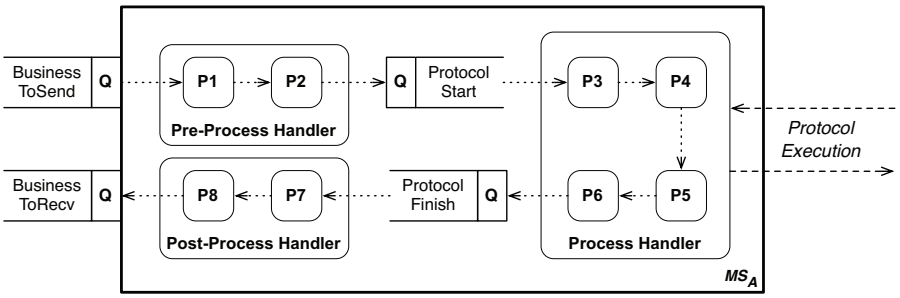

Fig. 4. An example of the messaging service configured with three handlers each with a message processor chain.

Figure 4 illustrates a particularly useful example configuration in which processors are partitioned according to their relationship to the execution of any protocol (i.e. pre-processing, processing and post-processing). This method of partitioning will be used to facilitate functionality discussed later in this section. This figure omits the use of signal and audit topics, their use is also discussed later in this section.

All handlers and their aggregate message processors are configured on a per service basis, this configuration for each instance also contains an identifier, assumed to be as unique as required, used to facilitate identification amongst collaborating parties.

3.2 Signal and Audit Topics

The previous section introduced signal and audit topics with regards to handlers and described their intended use. This design was intended to decouple the processors from event notification and auditing concerns (e.g. logging). In the case of the signal (or event) notification, processors simply emit their events to their encapsulating handler, these events are delivered to the signal topic, if specified, and interested participants are able to subscribe as desired.

For the audit topic, receiving verbatim copies of messages generated through execution allows logging to occur in a completely decoupled manner. Loggers may be created arbitrarily as subscribers to this audit topic, internally by editing the configuration of the messaging service or externally by simply subscribing to the topic.

It is intended that these topics may only be accessed by authorised parties to prevent vital information (e.g. non-repudiation evidence) being leaked to unauthorised subscribers.

3.3 Conversation Tracking and Awareness

The design so far illustrates how message processors are able to work on intercepted messages and how audit topics may be associated with handlers to provide logging. A vital capability in such a messaging service must be the ability to correlate groups of messages, particularly into which B2B conversation or specific protocol execution they belong. To solve this, we have identified a minimal set of tracking identifiers to support useful functionality:

Conversation Identifier This identifier is used to mark a message as belonging to a specific B2B conversation. All messages must contain this identifier.

Protocol Identifier This identifier is used to mark that a message is part of a specific protocol execution. Only protocol messages must contain this identifier, it makes no sense for business messages to contain this information.

Message Identifier This identifier is used to uniquely identify each message, all messages must contain this identifier.

These identifiers facilitate more useful lookups of logged messages and allow conversation and protocol executions to be tracked as they occur.

3.4 Deriving Technical from Business Requirements

As with the message classifications described at the beginning of section 3, it is useful to classify requirements in terms of business level and protocol (or technical) level. To be useful, a business level requirement should be declarative (i.e. specify the *what* rather than the *how*) but also have a complete mapping into technical requirements capable of satisfying it.

The pre- and post-processing phases of protocol execution as described in section 3.1 provide an ideal place within the messaging service to perform the transformation between these business and technical level requirements. The pre-processor will take business requirements and, using configuration and available knowledge, derive technical requirements satisfiable by the service. The post-processor will transform these back into business requirements allowing the underlying technical concerns to remain hidden from the enterprise.

Using accountability as the major example, we require only a single requirement at the business level. This must answer the question “Is accountability required for this individual message?”. For a business message where this is required, the necessary technical requirements and data can be derivedand used to execute the Coffey-Saidha protocol as discussed in this paper, the following data is required to facilitate protocol execution:

Non-repudiation Protocol Which specific non-repudiation protocol is to be executed.

TTP Identities The TTP and TSA to be used in the non-repudiable exchange must be explicitly identified.

Sender and Recipient To sender and recipient in the non-repudiable exchange must be explicitly identified.

An advantage of performing these transformations in the pre- and post-processing phases is that default technical requirements may be injected where none are specified at the business level. This allows a service to be configured to execute fair exchange unless instructed otherwise, for example.

A significant concern for these declarative annotations, however, is their descriptiveness and interpreted meaning, this is discussed in section 5.3 regarding future work.

3.5 Accountability as an Example

Section 3.1 and figure 4 demonstrated the partitioning of message processors based on classifying their role as a pre-processer, processor or post-processor with regards to protocol execution. This partitioning allows the design to achieve accountability through the use of non-repudiation implemented across three message processors named *NRPreProcess*, *NRPostProcess* and *NRProcess*. Each of these processors to be located within their respective protocol phase handler (pre-, post- and process) as shown in 5 and described in detail thereafter.

It is important to note that while intermediate locations between handlers are not shown in figure 5, a line directly connecting two handlers (e.g. 'Pre-process' to 'Protocol') implies the existence of one.

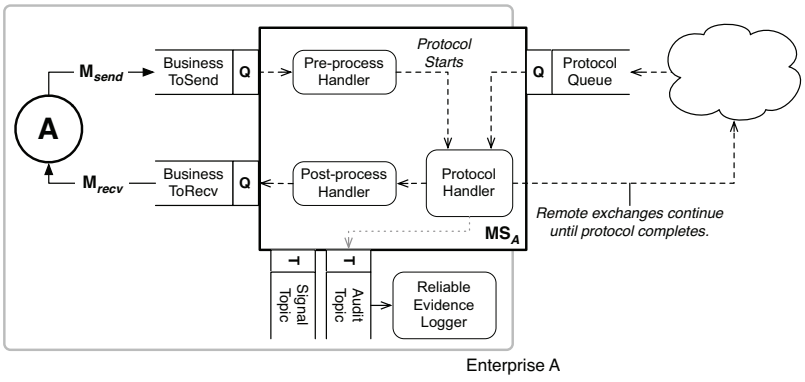

Fig. 5. Handler configuration inside MS_{A} supporting NR protocol execution.

NRPreProcess: This processor is responsible for checking each intercepted message to see if accountability is required and, if so, to derive the technical requirements and meta-data discussed in section 3.4.

NRProcess: Once the necessary technical information has been derived, this processer is responsible for executing the correct non-repudiable exchange protocol. An important note is that the output location for the NR processing handler will only be used when a single run of an invoked protocol has completed, additional procotol messages are likely to be exchanged with other participants (e.g. TTP, TSA and the collaborating participant) before this completion occurs.

NRPostProcess: Post-processing is the natural counterpart to the pre-processing. For accountability this means taking the result of a protocol execution and transforming it back into a business message for the recipient, to maintain transparency.

These three processors facilitate the execution of fair non-repudiable exchange protocols, the final requirement for fair non-repudiable exchange is the maintenance of an irrefutible audit trail. This is demonstrated in figure 5 by the subscription of a reliable evidence logger to the audit topic, which is connected to the protocol handler. This logger will receive copies of all protocol messages, including those containing digitally signed, and trusted non-repudiation evidence.

This modular design means that a messaging service needs only be configured with pre- and post-processing handlers if it is to deal with business level messages, their function of transforming between business and protocol message ensures clients are only ever exposed to business messages, even if fair exchange occured at the protocol message level.

A useful result of this is that NR TTPs and TSAs, who are only ever concerned with protocol level messages, may be configured as instances of the messaging service containing only a protocol handler.

3.6 Handling Business and Technical Level Failures

As with both requirements and messages, it is useful to define failures in terms of business level and technical level.

A business level failure includes scenarios such as failure to process (or respond) a message before a specified deadline or a business message being semantically invalid, for example. Were the messaging service in this paper to include a knowledge set comprising domain specific message exchange patterns (e.g. specific RosettaNet PIPs), generalised message exchange patterns (e.g. one and two-way asynchronous interactions in the RosettaNet Implementation Framework) and ongoing conversations it is feasible that the service could take automatic action to attempt to compensate with specific business failures (e.g. re-delivery in the case of a missed deadline). As it stands, the service opts to allow the business level clients within an enterprise deal with the detection and reaction

to these situations, this situation can be improved on by the discussed related and future work.

Technical failures present an altogether more complex issue to deal with, the struggle comes in dealing with them appropriately while attempting to maintain transparency to the business level clients within an enterprise. There may be times when a technical failure should be made known to a business client (e.g. fair non-repudiable exchange will never be possible with this participant) but doing so requires the enterprise to understand the underlying technical complexities, it may also be possible to use the signal topic to counter this issue. The messaging service could opt to attempt no automatic recovery in such situations and hope a business failure is eventually raised, or attempt to raise a business failure itself. The former is far from desirable and the latter would again require knowledge of the possible and ongoing conversations to invoke the correct business level conversations.

4 Related Work

The concept of supporting business conversations using intermediares in middleware is an explored approach, works relevant to this paper, are described in this section.

In 2007, Molina et al described an approach to guaranteeing consistent outcomes for B2B conversations carried out using message oriented middleware [8]. The approach provides synchronisation at both the protocol level and at the business level to collate individual outcomes and calculate the actual global outcome. This can be used to prevent inconsistent states occuring at the business level and alleviate the need for potentially slow and expensive business level compensation to be executed. The approach could be integrated into the messaging service although its use would mandate the deployment of the messaging service within each enterprise, this is discussed more in section 5.3.

In 2009, Strano detailed the design and implementation of support for specification and monitoring of electronic business contracts [15]. This is facilitated by a centralised monitor being fed events from participating business clients, using these events and specified contracts the monitor can decide whether the participants are meeting their obligations to contracts or infringing upon the rights of others. The messaging service described in section 3 could easily be adapted to transmit events based on the signal and audit topics to this centralised monitor, and use its output to better support governance of B2B conversations.

A possible issue with this work is that the events being fed to the centralised monitor may be generated by the participants themselves and thus, be untrustworthy. This issue can be directly addressed through the use of non-repudiation evidence. If, at the end of each B2B conversation, we enforce the non-repudiable exchange of the final message with the caveat that the *NRO* and *NRR* contain a hash of every message this provides evidence that is both trusted and cryptographically bound to the entire conversation. Both participants could use this to provide some form of trusted input to be used as evidence to the centralised

monitor. The messaging service described in this paper is capable of generating this evidence, but no provisions are made for interacting with the centralised monitor.

In 2006, Cook et al described a Web-Services specification for fair non-repudiation [3], providing a set of standards that allow individual messages to be intercepted transparently and exchanged in a non-repudiable manner, this work focuses solely on non-repudiation as the business requirement, providing multiple protocols and extension points for other non-repudiation protocols to be used. This work allows business level validation of messages to feed directly into the non-repudiation protocol phase meaning semantically and syntactically invalid business messages can be used to prevent unnecessary exchanges taking place, something this paper has not considered.

In 2004, the FIDES project described a complete version of their e-commerce security solution including the capability for fair non-repudiable exchange within private communities [9], the capabilities in this system are dependent upon the use of a specific client into which business documents are loaded and the recipient(s) specified, while this system is solves the issue of ensuring a recipient is capable of executing fair non-repudiable exchange, it is far from ideal as the client is completely opaque to the user and also means automating with existing B2B conversation specifications and implemented services becomes a non-trivial issue.

In 2007, Parkin et al provided an implementation using JMS to deliver voluntary non-repudiation on business messages being exchanged [12]. The approach was similar to that taken in this paper, the use of a gateway to intercept all business messages and provide voluntary non-repudiation on them all transparently by attaching and extracting digital evidence from outgoing and incoming business messages respectively. The approach described in this paper can be seen as a direct extension of this work, providing a more feature rich gateway (e.g. arbitrary processor implementation and conversational tracking) and also rendering strongly fair non-repudiation in addition to voluntary.

5 Results, Conclusions and Future Work

5.1 Resulting Implementation

The design shown in section 3 was implemented using JMS as the message oriented middleware. This use assumes all messages are JMS messages, allowing all requirements to be specified as *key-value* property pairs where the value is permitted to be any primitive Java type, or a *String* object. There were some minor caveats to overcome including a lack of wire-format, no standardised URI scheme [7] and no cryptographic primitives in the JMS API, enterprise design patterns were used to construct vendor independent solutions to these issues and allow the functionality to be delivered as designed.

The prototype has been tested and works successfully for two correctly behaving instances of the messaging service exchange messages in a fair non-repudiable manner.

5.2 Conclusions

The work presented in this paper is a design engineered towards supporting the satisfaction of declarative business level requirements. Accountability was used as the major example and satisfied through the execution of fair non-repudiation protocols. The design is flexible enough, allowing additional business and technical requirements to be supported transparently and maintaining compatability with existing standards including RosettaNet and ebXML.

The design and approach taken in this paper do have some serious ramifications however, first and foremost is the the compatability of both the declarative requirements and the way in which they are satisfied with the other partners. This paper (and the design in it) have adopted a very specific meaning for accountability and a way to satisfy that, using the Coffey-Saidha protocol. It would be unreasonable to assume all possible participants deploy our messaging service, meaning that accountability may mean something entirely different (or nothing at all) to another party. A possible solution to this is that the messaging service brokers with participants to attempt to arrive at some mutually acceptable definition or determines protocol support and continues to interact where possible.

Another significant issue is the handling of business and technical failures, while they have been considered, they have not been addressed in a particularly graceful or satisfying manner.

5.3 Future Work

Immediate future work would include improving the capabilities of the service, particularly adding compatability with the consistency support described in section 4. Integration with the centralised monitor as described in the related work section would be achievable by creating a message processor within the messaging service to generate and supply the trusted events as required. At the non-repudiation protocol level the Coffey-Saidha implementation requires the inclusion of abort and error sub-protocol support to be truly useful, implying the need for support of deadlines and timeouts.

The example requirements specified in this paper are done so on a per message basis, the issue of requirements on conversations has not been explicitly addressed but could be easily facilitated using knowledge about possible conversations and the requirements they have specified.

Further work may include integration with an Enterprise Service Bus (ESB), completely removing the concern for underlying message format and allowing the system access to intercept all outgoing and incoming messages of any type.

The integration of a rules engine in the messaging service could support predicates over conversations, messages and links (i.e. knowledge about all exchanges that have occured with a specific business partner). These predicates could possibly be translated from business requirements to further alleviate technical concerns for the business client.

As more declarative requirements are supported, it may become desired to provide an interface on to the messaging service allowing a plain business message to be passed in and business level requirements specified as parameters rather than just exposing a queue or topic. There is an obvious tradeoff here between maintaining transparency versus providing a possibly more descriptive interface.

References

1. Tom Coffey and Puneet Saidha. Non-repudiation with mandatory proof of receipt. *Computer Communication Review*, 26:6–17, 1996.
2. Nick Cook. *Middleware Support for Non-repudiable Business-to-Business Interactions*. PhD thesis, School of Computing Science, Univerisy of Newcastle upon Tyne, 2006.
3. Nick Cook, Paul Robinson, and Santosh Shrivastava. The rigorous implementation of a fair exchange protocol for non-repudiable web service interactions. *International Journal of Cooperative Information Systems (IJCIS)*, 4:565–597, 2006.
4. Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.
5. Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25:1606–1621, 2002.
6. Olivier Markowitch, Dieter Gollmann, Steve Kremer, and Université Libre De Bruxelles. On fairness in exchange protocols. In *In Proc. 5th Int. Conf. on Information Security and Cryptology (ISISC 2002), Springer LNCS 2587*, pages 451–464. Springer, 2002.
7. R. Merrick, IBM, P. Easton, D. Rokicki, Software AG, E. Johnson, and TIBCO. Uri scheme for java messaging service. Internet RFC JMS URI 05, November 2008.
8. Carlos Molina-Jimenez, Santosh Shrivastava, and Nick Cook. Implementing business conversations with consistency guarantees using message-oriented middleware. In *EDOC '07: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, page 51, Washington, DC, USA, 2007. IEEE Computer Society.
9. Ra Nenadic, Ning Zhang, and Stephen Barton. Fides - a middleware ecommerce security solution. In *in Proc. 3rd European Conf. on Inf. Warfare and Security (ECIW)*, pages 295–304, 2004.
10. OASIS. OASIS ebXML technical architecture specification, 2001.
11. OASIS. OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features, 2007.
12. Simon Parkin, David Ingham, and Graham Morgan. A message oriented middleware solution enabling non-repudiation evidence generation for reliable web services. In *ISAS '07: Proceedings of the 4th international symposium on Service Availability*, pages 9–19, Berlin, Heidelberg, 2007. Springer-Verlag.
13. RosettaNet. RosettaNet Implementation Framework: Core Specification, 2002.
14. RosettaNet. RosettaNet Overview: Clusters, Segments and PIPs. Technical report, 2009.
15. Massimo Strano. *Contract Specification for Compliance Checking of Business Interactions*. PhD thesis, School of Computing Science, Univerisy of Newcastle upon Tyne, 2009.
16. Jianying Zhou and Dieter Gollmann. A fair non-repudiation protocol. pages 55–61. IEEE Computer Society Press, 1996.