Analyzing Contract Robustness through a Model of Commitments

Amit K. Chopra[1], Nir Oren[2], Sanjay Modgil[3], Nirmit Desai[4], Simon Miles[3], Michael Luck[3], and Munindar P. Singh[5]

[1] Università degli Studi di Trento
akchopra.mail@gmail.com
[2] University of Aberdeen
n.oren@abdn.ac.uk
[3] King's College London
{sanjay.modgil, simon.miles, michael.luck}@kcl.ac.uk
[4] IBM India Research Laboratory
nirmit123@in.ibm.com
[5] North Carolina State University
singh@ncsu.edu

Abstract. We address one of the challenges in developing solutions based on multiagent systems for the problems of cross-organizational business processes and commerce generally. Specifically, we study how to gather and analyze requirements embodied within business contracts using the abstractions of multiagent systems.

Commerce is driven by business contracts. Each party to a business contract must be assured that the contract is *robust*, in the sense that it fulfills its goals and avoids undesirable outcomes. However, real-life business contracts tend to be complex and unamenable both to manual scrutiny and domain-independent scientific methods, making it difficult to provide automated support for determining or improving their robustness. As a result, establishing a contract is nontrivial and adds significantly to the transaction costs of conducting business. If the adoption of multiagent systems approaches in supporting business interactions is to be viable, we need to develop appropriate techniques to enable such software to reason about contracts in relation to their robustness.

To this end, we propose a powerful approach to assessing the robustness of contracts, and make three main contributions. First, we demonstrate a novel formal model for contracts that is based on *commitments*. Second, we define rules to evaluate the robustness of contracts. Third, we offer a methodology for modelling contracts to enable checking them for robustness. We validate these contributions via a study of real-world contracts.

1 Introduction

When agent-oriented software engineering (AOSE) first emerged, it developed a rich panoply of concepts, abstractions, and techniques based on the notion of agents and allied notions such as roles, protocols, organizations, and commitments. These notions address the inherently interactive nature of multiagent system and provide the key basis both for developing software applications that involve autonomous and heterogeneous participants and for distinguishing AOSE as a technical discipline from the rest of software engineering. The applications that AOSE is geared to addressing include cross-organizational business processes and commerce in general. These have clearly gained in social importance in the last decade or so since AOSE has been practiced. The needs that they bring up, especially of flexible modeling and enactment, and of managing complexity continue to speak to the importance of AOSE as a discipline.

However, as both traditional software engineering and AOSE have grown, it has become more and more important to bridge the gap between the two. In some cases, researchers have sought to use the tools and techniques of traditional software engineering to enhance AOSE, including efforts in programming tools and methodologies and in formal methods. Examples of the former include several works such as those surveyed by Nunes et al. [14], and examples of the latter include works by Meneguzzi et al. [13] and Telang and Singh [19, 20]. In other cases, researchers have sought to formalize concepts that originate in AOSE in ways that might influence traditional software engineering. Examples of this are works by Bordini and colleagues [2, 3] and by Weyns et al. [21]. The above works demonstrate the expanding overlaps between agent-oriented and traditional software engineering. However, they also demonstrate an interesting limitation in that generally the forays of AOSE into traditional practice so far take place in the later stages of design and development: either in the formalization of system specifications or in the development of executable or nearly executable software artifacts, and in their verification and validation.

The present work addresses one of the least understood and hence riskiest phases of the software engineering life cycle, namely, the determination and analysis of system requirements in the first place. Not only is the requirements phase the riskiest, it is also one where (for problems involving commerce, in particular) multiagent systems concepts can apply naturally and potentially facilitate the later phases.

Another novelty of the present work is that it takes what one might understand as a hybrid approach. It adopts the idea of commitments from AOSE as its key organizing principle and uses it to present a generalized model of business contracts in terms of a variety of commitments. This model sustains both (1) a methodology for identifying various types of commitments from traditional text-based contracts and (2) an approach for assessing the robustness of such contracts from the perspective of any of the parties involved. In other words, the present work seeks to incorporate AOSE concepts into the heart of traditional software engineering practice, seeking neither to replace traditional practice with AOSE concepts and technique nor merely to place AOSE concepts in a thin veneer on top while leaving the rest unchanged.

A business contract specifies the terms under which the contracting parties exchange services. In this context, a contract is *robust* for a party if it satisfies that party's goals and preferences. In general, practical contracts can be quite complex, usually because each party inserts clauses to protect its own individual interests. The question of whether such a contract is robust is an important one that is not trivial to answer. In fact, the robustness of a contract may be assessed in different ways. For example, whereas a contract may specify that a particular service will be provided, it need not specify *how* the specified service will be provided, leaving open the possibility that the method may be inappropriate in the eyes of some party. Alternatively, a contract may specify exactly what and how a service should be provided, but make no provision for rectifying problems when the service fails to be delivered due to accident or malice.

Two aspects of the complexity of contracts makes ensuring robustness difficult. First, traditional contracts are not explicitly structured according to a suitable high-level formal model. Second, the free text form of today's contracts complicates analyzing their content in any automated way. Multiagent systems offer promising solutions to help manage business relationships and enact business processes; however, without first assessing the robustness of contracts, agents cannot be relied upon to agree upon or execute contracts of real significance.

In this paper, we provide an approach to modeling contracts specifically in order to address the problem of unambiguously analyzing their robustness. We treat a contract as a set of interrelated *commitments* among those parties who have signed it. These commitments play differing, interconnected roles in the overall contract and support a formal analysis to determine potential threats to the robustness of the given contract. For example, robustness is enhanced when a commitment to provide a service occurs with a concomitant commitment to resolve problems in cases where that service could not be delivered.

We make three contributions that together address the problems of how to *analyze* contract robustness, and how to *design* contracts to ensure that they are robust. First, we provide a structured *model* for expressing contracts. Second, we outline a *methodology* for identifying the various kinds of commitments that occur in a textual contract. The methodology helps formalize contracts in terms of the structured model. Third, we specify *rules* that can be applied to the model in order to determine the robustness of the underlying contract. We motivate our approach using examples from a real contract; we validate our approach against another real contract.

The rest of the paper is organized as follows. Section 2 introduces our running example. Section 3 describes our structured, commitment-oriented model of contracts. Section 4 outlines our methodology for expressing human-oriented contracts in our model. Section 5 specifies rules governing robustness of contracts. Section 6 provides an evaluation using a second case study, and Section 7 discusses related work. Finally, Section 8 concludes with discussion of future directions for research.

2 Motivating Example

To illustrate and evaluate our approach, we consider a real-world services contract between Advanced Semiconductor Engineering (ASE) and Motorola.[6] The contract is for the assembly and testing of semiconductor chips, and the provisioning of related services. To save space, we describe only its relevant snippets. The preamble identifies the parties and their motivations for entering into the contract, as the following snippet shows.

> **Preamble:** MANUFACTURING SERVICES AGREEMENT... WHEREAS, Motorola and ASE desire to establish a strategic supplier relationship in which

[6] http://contracts.onecle.com/ase/motorola.mfg-korea.1999.07.03.shtml

> ASE will utilize the capacity at its final semiconductor manufacturing operation and facilities of ASE Korea located at Paju, Korea (the "PAJU FACILITY") on a priority basis to perform the assembly, test, and associated services on certain semiconductor products for Motorola.

The contract includes distinct sections, each grouping clauses that impose interrelated demands on the contracting parties. ASE will use its facility in Korea to assemble and test semiconductor products (the *contract products*) for Motorola. Motorola will provide the requisite specifications and equipment to enable ASE to carry out its task. Motorola will also provide monthly forecasts to aid ASE in capacity planning. Motorola will place purchase orders with ASE for the contract products, upon which ASE will ship the products to destinations specified by Motorola. ASE will then invoice Motorola for payment according to the prices agreed upon in the contract. Clauses in the contract also cover concerns such as insurance, indemnity, liability, and so on.

3 A Model for Robust Contracts

A contract is robust in fulfilling agents' goals under varying circumstances primarily because of the *commitments* made in the contractual clauses. It is from these commitments that we can assess what to expect from agents executing the contract. Therefore, the first step in our approach is to provide a basic model for contractual commitments. We go on to describe how robustness can be defined in terms of such commitments, and then provide an enhanced commitment model specifically designed to model contractual information relevant to analyzing robustness.

3.1 Background: Commitments

The expression C(DEBTOR, CREDITOR, CONTEXT, *antecedent*, *consequent*) means that the debtor commits to the creditor for bringing about the consequent provided the antecedent holds. In contractual terms, a commitment represents a proposed business exchange: the antecedent and consequent represent the considerations of the creditor and debtor, respectively.

Importantly, a commitment arises within a context, which captures the legal, social, or community setting in which the commitment is enforced. A subtle feature of our approach is that here the context can correspond to either a real-life institution or organization, such as eBay or the European Union or the famous fish market of Blanes [16]. A context is an active entity and can be modeled as an agent in its own right: a context in this sense imposes regulations on the participants, and it might penalize or eject noncompliant participants. The context itself may or may not have any consideration in the business exchange; its primary function is regulation. Often, the context plays the role of an arbiter in disputes. Within a contractual setting, the context typically consists of the legal framework under which the contract is signed, together with the domain ontology and the contract document itself. In other words, given a certain legal system, an understanding of the world, and a contract (all of which make up the context), certain commitments between the contracting parties arise and are manipulated in a natural manner.

More specifically, a contract is a set of commitments, each of which has the same context. As an example, an ordering process may involve two commitments: c_1 = C(SELLER, BUYER, ORG, *pay*, *shipGoods*) and c_2 = C(BUYER, SELLER, ORG, *buyGoods*, *pay*). Here we use ORG as the context within whose scope the contract takes place. The first commitment requires the seller to ship the goods to the buyer once payment has been made, whereas the second commits the buyer to pay for goods it has purchased. Notice that the BUYER and the SELLER may themselves be organizations, each with its own internal structure.

A key benefit of the commitments representation is that commitments can be manipulated in a perspicuous and principled manner, thus yielding the flexibility needed in automated contractual interactions. A commitment may be *created*. When its antecedent holds, it is *detached* meaning that it reduces to a commitment to bring about the consequent unconditionally. When its consequent holds, it is *discharged*—this could even happen before the commitment is detached. The creditor may *assign* a commitment to another agent. Conversely, a debtor may *delegate* a commitment to another agent. A debtor may *cancel* a commitment and a creditor may *release* the debtor from the commitment.

Note that the debtor and creditor of a commitment need not be its direct performer or beneficiary [17]. Often, each party would play a role in a participating organization, and would represent the interest of the organization for the purposes of the commitment. For example, a manufacturer may commit to repairing some piece of machinery for a factory, but the repairer may be a subcontractor of the manufacturer.

3.2 Robustness of a Contract

The robustness of a contract depends on how its commitments relate to the goals of the contracting parties.

Definition 1 relates each of a party's goals to commitments in the contract. It says that the fulfillment of a subset of commitments—in any manner—must lead to the satisfaction of the goal, that is, the goal is *supported*. The set of commitments leading to fulfillment of the goals may represent either the normal way to fulfill the goals where all services are delivered successfully, or a *compensating* way to fulfill the goals where some commitments are violated but compensating commitments are fulfilled.

Definition 1. *A contracting party's goal is* supported *by a contract if and only if the fulfillment of the subset of contract commitments, in which the party is the creditor, entails the goal.*

Given the above definition, we can then define what it would mean for a contract to be robust for a contracting party.

Definition 2. *A contract is* robust *for a contracting party if all of the contract party's goals are supported by the context. A contract is* robust *overall if it is robust for all its contracting parties.*

In order to specify how to assess robustness, we must define what it means for (1) a contracting party's goals be *entailed* by the contract and (2) a commitment to *compensate* the failure of another commitment. Both of the above relate to the different types

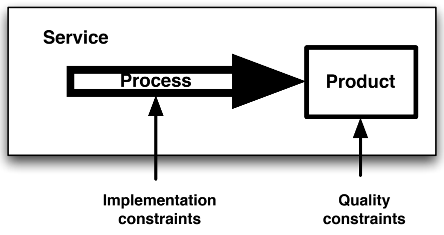

Fig. 1. Control flow for the reasoning process.

of behavior a contractual commitment can address. Therefore, it is important to model the kinds of commitments depending on the purpose they serve in the contract. Below, we enhance our basic commitment model to include the specification of commitments based on their purpose.

3.3 Enhanced Commitment Structure

From our examination of real-life contracts, we observe that the commitments occurring within them exhibit a particular structure, which we exploit to assess the robustness of contracts.

At the heart of this structure is the idea of a *service*. A service is the creation of some *product* by a *process* under the *assumed circumstances*, as shown in Figure 1. The product is what an agent actually wants, whereas the process is the means by which the product is brought about. The product may be an artifact or an activity taking place or something holding true about the world. Significantly from the perspective of robustness, it is often the case that a product can be evaluated by its consumer whereas the process is usually hidden. The *assumed circumstances* constitute normal, expected operation: a contract sets up expectations about what each party will do and does so assuming the rest of the world works in a particular way. Considering these assumed circumstances enables us also to consider what should happen when they do not hold in some way.

We view contracts as inherently symmetric among the parties. Thus each party potentially provides one or more services to the others. A *service commitment* is, then, a commitment whose debtor plays a role in which it provides a service to the creditor of the commitment. A service commitment states what is to be produced by the service and under what assumed circumstances, without further describing the product or process. In terms of the overall structure of a commitment described in Section 3.1, the service product is the consequent of the commitment.

A contract contains a set of service commitments. For each service, there are then a number of other constraints and commitments that are meaningful when understood in context of the service.

- *Quality constraints*, with regard to a service, are restrictions on the debtor to ensure that the service product is of a minimum acceptable quality.

– *Implementation constraints*, with regard to a service, are restrictions on the debtor to ensure that the process used for production meets certain requirements.
– A *contingency commitment*, with regard to a service, is a commitment on the debtor or a third party to provide an *alternative service* when the assumed circumstances do not hold (and stated *contingency circumstances* hold instead).
– A *resolution commitment*, with regard to a service, is a commitment on the debtor or a third party to provide an *alternative service* when the service commitment is violated.
– An *audit commitment*, with regard to a service, is a commitment on the debtor or a third party to perform an audit of the service, the product of which is the record of the service having been conducted.

Using the above enhanced structure, we model a contract as a set of such commitments. The structure for documenting a commitment C is shown in Table 1. As explained in Section 3.1, each contract has a CREDITOR and a DEBTOR agent. The *antecedent* is divided into an *Activation* condition, which states what triggers the commitment to apply, and *Assumed* circumstances, which states what is assumed to hold when the commitment applies. Both must be true for the commitment to apply, but they are dealt with in different ways. If the *Activation* condition does not hold at some time, then the commitment simply does not apply at that time. Conversely, if the *Activation* condition holds but the *Assumed* circumstances do not, then the *Contingency* commitment applies instead (if one is given).

The *consequent* is similarly divided into parts: for the *consequent* to be true, the *Product* must have been produced such that the *Quality* properties hold true of the service product and the *Implementation* properties hold true of the service process.

Each commitment C additionally has related commitments. A *Resolution* commitment is applicable when the original commitment C is violated, that is, the antecedent of the resolution commitment is the violation of the original commitment. An *Audit* commitment is applicable whenever commitment C's process is enacted (and thus will has as antecedent the same or a more general antecedent than commitment C) and produces documentation regarding the service process.

A contract modeled so as to analyze robustness then, is a set of enhanced commitments, EC(CREDITOR, DEBTOR, *activation*, *assumed*, *product*, *quality*, *implementation*), together with functions that map from enhanced commitments to resolution, contingency, and audit commitments (each of which themselves is an enhanced commitment).

4 A Methodology for Contract Robustness

Given this structured model for expressing contracts in a way that is appropriate to analyzing their robustness, we are now able to present a methodology for determining whether a contract is robust or not. Our proposed methodology has two stages: first, it involves mapping the contract text to the commitments model introduced in Table 1; and, second, it involves applying rules to this mapping to check for robustness.

The first phase of our methodology consists of a number of steps, with each step identifying certain artifacts within the contract, and verifying whether these artifacts

Table 1. Enhanced commitment structure to assess robustness.

Enhanced commitment	
Reference	*An identifier to refer to the commitment*
Creditor	*The beneficiary of the service*
Debtor	*The party responsible for providing the service*
Antecedent	
Activation	*Under what circumstances this commitment applies*
Assumed	*Circumstances assumed in providing service*
Consequent	
Product	*The product of the service*
Quality	*The properties that should hold for the product*
Implementation	*The properties that should have held for the service process*
Related	
Contingency	*A commitment regarding what should be done when the assumed circumstances do not hold (referred to by identifier)*
Resolution	*A commitment regarding what should be done when this commitment is violated (referred to by identifier)*
Audit	*A commitment to produce data about how this service is performed (referred to by identifier)*

meet some basic rules to ensure the contract is correct and robust in trivial ways. For example, verifying might mean ensuring that no commitment has the same creditor as debtor, and that it is clear when the contract begins and ceases to *have force*. In the next section (Section 3.2), we introduce the more rigorous robustness rules, which may not hold even for apparently well-drafted contracts.

Our methodology consists of the following steps. For each step, we give the number of the section in this paper in which that step is explained.

1. Identify the critical entities involved in the commitments (Section 4.1)
   (a) Identify the contracting parties (Section 4.1)
   (b) Identify each contracting party's goals (Section 4.1)
   (c) Identify domain concepts (Section 4.1)
   (d) Identify contract scope (Section 4.1)
2. Map the above entities into the commitment model (Section 4.2)
   (a) Model services, processes, and products (Section 4.2)
   (b) Model commitments regarding services (Section 4.2)
3. Check the robustness of the commitments (Section 5)
   (a) Check that the contract meets each party's goals (Section 5.1)
   (b) Check that it is well specified how services should be provided and how to handle circumstances in which the services are not provided as specified (Section 5.2)
   (c) Check that the contract does not place conflicting demands on the parties (Section 5.3)

We illustrate the methodology via clauses selected from the ASE-Motorola contract, especially an abbreviated form of Clause 11.

Clause 11: ASE shall ship the Contract Products to the destinations identified by Motorola. Motorola shall acknowledge to ASE the receipt of each shipment of Contract Products, stating the quantity and type of, and any damages existing at delivery to, such Contract Products within [X days] of receipt at Motorola's ultimate destination ... ASE shall certify to Motorola with each shipment that the Contract Products contained therein have successfully passed applicable testing and meet all specifications ... If Motorola rejects any Contract Products, Motorola and ASE shall confer to determine the reason for the rejection. ASE shall immediately exercise commercially reasonable efforts to develop and implement a corrective action plan for any errors, including manufacturing errors or defects, identified in its systems.

4.1 Entity Identification

It is crucial to identify the various artifacts referred to in the contract. These artifacts may then be used within commitments in some structured or unstructured manner. In the former case, rules may be created identifying how they may, or should, be used in order to lead to a robust contract. The following entities are of interest.

Contracting parties A contracting party named by the contract is an entity whose commitments and responsibilities are described by the contract, and who is a signatory to the contract. In Clause 11, ASE and Motorola are the contracting parties.

A contract may identify specific roles within a contracting party, when it is an organization. For example, ASE is committed to providing *Motorola Employees* with office facilities according to Clause 5 (not shown). Other roles mentioned in the contract include those of a *coordinator* and the *ASE account team*, which then includes additional roles such as *manager* and *executive*.

Contract goals Business parties adopt a contract if it is conducive to achieving their goals—if the contract is robust, then these goals will be achieved. The preamble specifies the overarching goal; here, this is the successful production and delivery of semiconductor products from ASE to Motorola. This leads to other identifiable subgoals regarding high-level concepts such as the goal of having ASE deliver the product in a timely manner, the defect rate falling below some threshold, and so on. As we discuss below, each of these goals must be satisfied by some combination of commitments specified in the contract.

Domain concepts Contracts specify what the contracting parties are committed to do within some domain, specifying the relevant states of domain artifacts and how to manipulate them. Domain concepts in Clause 11 include *products*, *rejection*, *receipt*, *destination*, *damage*, among others. Although it is beyond the scope of this paper, we assume a suitable ontology for each domain.

Scoping A robust contract should specify when it is in effect, and when it expires, for example, via a termination clause that specifies the conditions under which the contract ends. Clause 3 (not shown) within the Motorola-ASE contract states that the contract is effective from the signing date, and is in force for five years. It also provides alternative ways of terminating the contract early. A basic requirement of robustness is clarity of the scope.

Basic Rule 1 *A robust contract specifies the conditions when the contract begins and ends.*

4.2 Mapping to Commitment Model

Once we have identified the critical entities, we map them into our commitment-based model.

Services, processes, and products Clearly, it is necessary to identify the services to which the contractual commitments apply. For each service, its product—that is, its desired outcome—must also be identified. Each service is expressed, or sometimes implied, in contract clauses using the identified domain concepts, and each party's goals are expressed in terms of the services.

The *Preamble* in our example contract describes the primary services under consideration, as follows: "the assembly, test and associated services on high quality semiconductor products in volume." This hints at a service whose product is assembled semiconductor products and a service whose product is tested semiconductor products. Later clauses identify other "associated" services. For example, Clause 11 includes "ASE shall ship the Contract Products to the destinations identified by Motorola," the product of which is the delivery of goods, and goes on to make statements about how this service should be provided.

Service commitments Because we view a contract as an aggregation of the commitments it imposes upon the contracting parties, determining whether a contract is robust involves identifying the commitments found in the contract. The remainder of the methodology focuses on these commitments, and the relationships between them.

Each service identified in the contract has a corresponding commitment, with one identified party as debtor, and another as creditor. It is a basic prerequisite for robust execution that any commitment *must* have some contracting party (and sometimes a specific role within it) as the commitment's debtor and creditor, implying the following rule.

Basic Rule 2 *A robust contract must ensure that every commitment within the contract will have a contracting party as a debtor and a creditor.*

For the primary shipment service referred to in Clause 11, the creditor is Motorola and the debtor is ASE. Further, a valid commitment must have distinct parties as debtor and creditor.

Basic Rule 3 *The same entity may not be named a debtor and a creditor within a single commitment.*

Finally, the given contract must translate unambiguously to our formal model, and so the following basic rule applies.

Basic Rule 4 *A commitment must only refer to concepts that have been defined within the domain ontology.*

Table 2. Service commitment for shipping.

Commitment for shipment service	
Reference	C11-Shipment
Creditor	Motorola
Debtor	ASE
Antecedent	
Activation	When products ready to ship
Assumed	
Consequent	
Product	Products arrived at specified Motorola site
Quality	No damage to products
Implementation	Perform applicable tests to certify products
Related	
Contingency	
Resolution	C11-Rejection
Audit	C11-Receipt
Audit	C11-Quality

Example We apply the above to the initial modeling of Clause 11. Table 2 expresses the commitment to perform the primary service of the clause, that is, shipment of products to specified destinations. This commitment is given an identifier, C11-Shipment, and refers to three other commitments extracted from the clause: C11-Rejection, C11-Receipt, and C11-Quality. For brevity, we omit the models for the latter two audit commitments; those refer to Motorola's commitment to provide a timely receipt for products received, and ASE's commitment to provide a statement of quality, respectively. The resolution commitment, C11-Rejection, is invoked when the service product is not achieved, the commitment to quality (no damage) is violated, or the commitment to implementation (tests performed) is not fulfilled.

Table 3 shows the model for C11-Rejection. Here, the service performed is the correction of the cause of rejection. No further commitment is involved, as the clause does not specify what should be done to audit the commitment or in contingency situations.

Table 3. Commitment to rectify problems (rejected products).

Commitment for acting in case of rejection	
Reference	C11-Rejection
Creditor	Motorola
Debtor	ASE
Antecedent	
Activation	Motorola rejects delivered products
Assumed	Within X days of delivery
Consequent	
Product Quality Implementation	Corrective plan of action developed and implemented by ASE
Related	
Contingency	
Resolution	
Audit	

5 Robustness Rules

Given the model of contracts in the preceding section, we now specify *rules* for determining the robustness of contracts expressed in that model. We divide such rules into the following main categories:

1. those that determine whether the contract contains the content required by each party;
2. those that determine whether each contract commitment is handled robustly; and
3. those that apply to consistency between commitments.

5.1 Necessity Robustness Rules

A robust contract must ensure that each contracting party's goals are satisfied when the contract executes correctly. The consequent of a service commitment may be used to capture the commitment's creditor's goals (when the commitment's antecedent holds). Therefore, the desired outcome of a contract may be captured by some subset of the contract's service commitments. A robust contract must thus satisfy the following rule.

Robustness Rule 1 *Each goal expected to be satisfied by the contracting parties should be (a necessary implication of) the consequent of a service commitment.*

Applying this rule to our example, the commitments shown in Tables 2 and 3 are judged robust with regard to this rule: on the former's completion, Motorola will have the components it desires; on the latter's completion, any problems will have been appropriately addressed.

5.2 Coverage Robustness Rules

A service commitment can often be fulfilled in multiple ways, and not all are of equal value to the contracting parties. In order to be robust, the contract must ensure that a commitment is met appropriately.

Robustness Rule 2 *Each service commitment must have corresponding quality constraints that specify what it means for the service product to achieve an adequate standard.*

Table 2 shows a simple statement of the quality required of the product: no damage should have occurred. In the commitment in Table 3, no quality constraints are given. Whereas this omission may be deemed appropriate by the contracting parties, the above rule highlights the fact that the contract is less robust if Motorola places no criterion on what an acceptable corrective plan can be.

Whereas the quality constraints concern the service product, we may also apply criteria for judging the process by which the service is conducted, leading to the following rule.

Robustness Rule 3 *If a service commitment may be met in a number of ways, a proper subset of which capture the creditor's goals, then the service commitment should have corresponding implementation constraints that specify what it means for the service commitment to have been achieved in a satisfactory manner.*

Table 2 shows a commitment by ASE to apply tests for damage and to ensure specifications are met prior to delivery (and therefore part of the service process). In contrast, Table 3 gives no implementation commitment. The above rule highlights the fact that the contract is less robust if Motorola places no criterion on what process is acceptable in developing a corrective plan, for example, the factors that ASE should take into account.

The fulfillment of service commitments and quality constraints is usually publicly observable. For example, whether ASE has manufactured the semiconductor chips up to the requisite standard is verifiable by Motorola once Motorola has received the chips. However, implementation constraints restrict the internal processes a contract party employs; compliance with such commitments is not visible outside the company. For example, Motorola cannot ascertain from outside ASE whether ASE has met the ISO 9000 standards in manufacturing the chips. Hence, implementation constraints call for audit commitments.

Robustness Rule 4 *Each service's implementation constraints* must *have a corresponding audit commitment that ensures that the satisfaction or violation of the constraints is detected.*

If a commitment has been violated (for example, if the product is not available, or if quality or implementation constraints haven't been followed), then the creditors' goals may not be achieved. In order to be robust, therefore, the creditor in the commitment requires that some compensating commitment comes into force.

Robustness Rule 5 *Each commitment* must *have a corresponding resolution commitment that ensures that the violation of the former commitment results in a suitable sanction on the debtor.*

Table 2 shows two commitments to ensure correct auditing by both parties involved. It is only by auditing that any violations of the implementation constraints are detected. There is also a resolution commitment, to specify what should be done when the product or process is inadequate according to the quality and implementation constraints. Table 3 shows no audit or resolution commitments are given. The above rules highlight the fact that the contract is less robust if there is no record of ASE having produced and implemented such a corrective plan, or what action to take if ASE fails to produce such a plan.

Further, for the debtors of a contract commitment, the contract is robust only if it adequately accounts for exceptional circumstances, beyond those assumed in normal operation. We ensure the robustness of the contracts in relation to these aspects, with the following rule.

Robustness Rule 6 *Each commitment may have corresponding contingency commitments that ensure that, in each exceptional circumstance envisaged, the violation of the former commitment does not result in an inappropriate sanction on the debtor.*

Table 2 shows no contingency commitments because the contract fails to specify assumed circumstances. The absence of assumptions should draw the modelers' attention, but may merely indicate that there is no contingency to consider. Table 3 also states no contingency commitment, but does have assumed circumstances. The above rule highlights the fact that the contract is less robust if it is not specified what should be done if Motorola only rejects a product long after (more than X days) it has been delivered.

It might seem that, if applied recursively, the above rules could lead to an infinitely large contract; for example, each commitment requires another commitment for resolution. However, our use of the context of a contract—as in a business contract within a wider legal system—provides a natural solution. Not all of the associated commitments mentioned in the rules above need to be in the contract document itself; many may be present in the wider context. Ultimately, the audit, resolution, or contingencies of contextual commitments may be captured via general approaches, such as "file a lawsuit."

5.3 Consistency Robustness Rules

The above rules consider the requirements of robustness on each commitment. The robustness of a contract as a whole depends in addition on whether its commitments are realizable.

It should always be clear to a contracting party what to do to fulfill the contract, even in the case of multiple failures. Further, if success in one commitment prevents success in another, then the contract cannot be robust. A particular example of this is where two commitments require the same party in the same system state to do two conflicting things. A robust contract does not have such conflicts between its commitments, and the following rule expresses this constraint.

Robustness Rule 7 *For any given contracting party and applicable system state, by performing an action necessary to avoid violating one commitment, the action should not violate any otherwise nonviolated commitment.*

Taken together, the rules specified above provide us with a means of ensuring that a contract is robust at the point of specification. The full set of rules is summarized in Table 4, indicating which aspects of the contract each rule applies to.

Table 4. Contract rules.

Rule	Target
BASIC RULE 1	Scope of contract
BASIC RULES 2 & 3	Services and contracting parties
BASIC RULE 4	Well-defined contract
ROBUSTNESS RULE 1	Product
ROBUSTNESS RULE 2	Quality constraints
ROBUSTNESS RULE 3	Implementation constraints
ROBUSTNESS RULE 4	Audit commitments
ROBUSTNESS RULE 5	Resolution commitments
ROBUSTNESS RULE 6	Contingency commitments

6 Evaluation

We used the Motorola-ASE contract as primary inspiration for our approach to modeling and assessing contract robustness (along with our prior experience with case studies as part of electronic contracting projects). To evaluate our proposed approach, we took an entirely independent contract and applied our methodology to it. Figure 2 shows an excerpt from a short contract[7] between a juggling society and an event organizer. We now show how our methodology applies to determine whether this contract is robust.

6.1 Entity Identification

The two contracting parties involved in this contract are the JUGGLING CLUB (UJC), and the CANTERBURY CENTRE DINNER (CCD). Additional roles include PERFORMER and JUGGLER. As we see below, this contract obeys Basic Rules 2 and 3.

The CCD's goal from the contract is to obtain performers for their dinner. The UJC's goal is to get paid.

Apart from temporal concepts (relating to dates and times), and general concepts such as money, we may identify the domain concepts listed in Table 5. Since only these concepts are referred to within the contract, Basic Rule 4 is satisfied.

The contract initiates as soon as it is signed and it is *implied* that it expires at the end of the performance. Note that the lack of an explicit expiration condition suggests

[7] http://users.ox.ac.uk/ juggsoc/contract.shtml

Contract For: Canterbury Centre Dinner 2003 (“CCD”),
Friday 6 June 2003, 24 High Street, Canterbury.

This agreement is entered into between the University Juggling Club (“UJC”) and the Canterbury Center Dinner 2003 on the following terms:

1. Service Provider: University Juggling Club.
2. Employer: Canterbury Center Dinner.
3. To be provided by UJC: Performers: J Woods (juggler); one other juggler; all equipment necessary for performance.
4. To be provided by CCD: Cloakroom.
5. Venue address: 24 High Street, Canterbury.
6. CCD understands that performances are restricted in venues with ceilings of insufficient height. The ideal height is 5 meters. Outside performances are restricted in rain or strong winds.
7. Date of Performance: Friday 6 June 2003, starting at 6:30PM.
8. Duration of Performance: 1.5 hours. Short (less than one minute) breaks are part of the performance.
9. Fee: £30 per juggler + £10 expenses + £90 insurance (total £160).
10. If UJC is forced to cancel, all monies (including £90 deposit) will be refunded in full. If the Employer cancels with at least 24 hours notice, UJC will retain £90 and return any other monies.
11. Should poor weather mean that the Event takes place indoors, UJC will refund £10 expenses.
12. Performers will not consume any alcohol until after completion of services as agreed.
13. CCD will be responsible for compensation to UJC for damage to equipment caused by those attending the Event unless damage is caused when (if) Performers have left equipment unguarded.
14. UJC will be liable for any injury sustained by a guest at the Event if such injury results from provision of services as agreed upon in this contract unless the Event fails to provide a suitable area for performance.

Fig. 2. An extract from a contract to provide juggling services.

one problem with the robustness of the contract. One may envision a situation where some equipment is damaged, and a disagreement arises as to whether this damage falls under the contract or not (for example, when the jugglers and guests are on their way home from the dinner). Thus, Basic Rule 1 is not satisfied within this contract.

6.2 Mapping to Commitment Model

We now map clauses from the contract to the commitment model. Clauses 3 to 7 imply a service to be provided: the provision of jugglers and equipment by UJC, modeled in Table 6. UJC is the debtor, CCD is the creditor and the eventual product of the service is that the jugglers perform at the event. Implementation constraints are specified: that

Table 5. Domain concepts for the *Juggler* contract, grouped according to the commitments that they most closely relate to.

Service	performance, breaks, guests, venue, equipment
Contingency	deposit, damage, injury, guarding, poor weather, cancellation
Implementation	alcohol consumption, cloakroom, performance area, indoors, height, outside
Resolution	compensation, liability, refund

the jugglers remain sober (Clause 12). Where the service cannot be provided due to poor weather conditions (assumed circumstances not holding, Clause 11) or the performance is canceled by UJC (violation of commitment under assumed circumstances, Clause 10), contingency and resolution commitments apply, respectively. Clause 9 is a commitment for a separate payment-for-juggling service, and so is not modeled here.

Table 6. Service commitment: provide jugglers, equipment (the numbers refer to clauses in the textual description).

Commitment for providing resources	
Reference	C-ProvisionOfResources
Creditor	CCD (2)
Debtor	UJC (1)
Antecedent	
Activation	Agreement to contract
Assumed	Venue is indoor and of adequate height or outdoor and there is no rain or strong winds (6)
Consequent	
Product	C-JugglerPerform (3,4,5,7)
Quality	C-PerformanceFor1.5Hours (8)
Implementation	C-JugglersWillNotConsumeAlcohol (12)
Related	
Contingency	C-PoorWeather (11)
Contingency	C-Cancellation (10)
Resolution	C-CompensationResponsibility (13)
Resolution	C-Injury-Liability (14)

6.3 Assessing Robustness

Having identified the commitments, we may check whether they meet the appropriate robustness rules. Clearly, each desired outcome of the contract meets the commitments specified in Clauses 3, 4, 7, 8 and 9, as a performance will take place, and UJC will be paid. Thus, Rule 1 is satisfied.

According to Rule 2, each service commitment must have associated quality constraints. Whereas one assessment of quality is given for the service in Table 6, and so the clause can be judged somewhat robust, other quality measures may also be considered (for example, specifying how capable the juggler should be).

UJC agrees to implementation constraints: that the jugglers do not consume alcohol while performing. Note that although there is no corresponding audit commitment, this is only because the contracted performance is slated to happen in a public venue and CCD would easily be able to detect noncompliance on part of the juggler. Thus Rule 4 is satisfied. The parties may consider additional implementation constraints, for example if there are any stipulations that should be made about how the product is reached, such as whether the organizers are given prior warning about when the jugglers will arrive.

There are some commitments for contingency and resolution in Table 6. Therefore, there is some robustness in this regard according to Rules 5 and 6. However, the contract can be even more robust if consideration is made of the other ways in which the assumed circumstances may not come about or the service is not provided. For example, the assumed circumstances are a conjunction of criteria and the contract does not say how to handle jugglers arriving at a venue with too low a ceiling. Similarly, the quality constraints require jugglers to remain sober, but there is no means of redress specified if this commitment is violated.

The juggling contract's inability to deal with such unexpected situations, together with its vagueness, means that it lacks robustness in several ways, and that in unexpected situations, disagreements between the parties may occur that the contract may be unable to resolve.

7 Related Work

Tropos is one of the leading AOSE methodologies with a substantial emphasis on early-stage requirements [4]. Tropos is centered on the notion of goals (along with dependencies among goals), and generally works best where the system-to-be is built to accommodate the goals of the stakeholders, modeled as actors. Tropos does not naturally apply to cross-organizational settings, where there is no unique system-to-be but rather one per stakeholder and where often the challenge is to specify the interaction *protocol* or rules of encounter rather than a complete implementation of a system. Mallya and Singh [12] relate protocols with Tropos using dependencies as bases for inducing protocols. Telang and Singh [18] have sought to incorporate commitments into Tropos as first-class modeling concepts. However, the existing works on Tropos have a general bias toward greenfield system designs whereas the approach we propose above begins from existing contracts and thus potentially can apply when a functioning (though potentially inadequately functioning) cross-organizational system is already in place.

Much work has been done on using automated contracts within computer science, and particularly within the area of multiagent systems. It is possible to categorize this work based on the contract life cycle. Our work in this paper concerns itself with the first stage of the contract life cycle, namely contract drafting. In this phase, themes such as the precise language used to represent the contract are important, as well as challenges such as contract negotiation (for example, as studied by Carbogim and Robertson [5]) and contract validation. Once a contract is drafted, it comes into effect, and further challenges such as contract monitoring and enforcement become important, but are not further discussed here. Daskalopulu et al. [7] describe logic-based tools for this end.

Many contract languages that have been proposed, including those by Abrahams and Bacon [1], Grosof and Poon [11], and Governatori [10]. However, none of them represent contracts as a set of commitments as we do. We do not study how a contract comes into being, concentrating instead only on whether it is robust or not. Thus, our work falls into the area of contract validation. However, most work on contract validation concerns itself with either ensuring that contract clauses are consistent, for example, by Daskalopulu [6], or ensuring that a sound legal basis exists, for example, by Gisler et al. [9]. The notion of robustness adds to, rather than replaces these concerns.

The only other large-scale analysis of contractual requirements that we are aware of is the work of Daskalopulu et al. [6], who investigate how to support large engineering contracts. However, their work was focused on identifying language requirements for such contracts, not on a software engineering methodology as here.

The work of Desai et al. [8] is relevant in this regard. Desai et al. study contracts from the perspective of utility theory as a basis for determining from the perspective of a contracting party whether a particular contract is *safe* (never produces negative utility) or *beneficial* (produces positive utility) for it. It would be useful to incorporate Desai et al.'s representation and reasoning approach into our methodology, although a practical challenge that such economic approaches face is determining the relevant utilities and probabilities in domains of sufficient complexity to be practically valuable.

8 Conclusions and Directions

In this paper, we have sought to develop a model, a methodology, and heuristic rules by which we can capture and analyze requirements from business contracts as a potential basis for developing robust multiagent implementations of software systems in open environments.

We identified the notion of *robustness* as critical to a contract. Informally, a robust contract is one that meets the contracting parties' goals for the contract, and handles unexpected situations gracefully. We proposed a methodology for determining whether a contract is robust, and evaluated portions of this methodology on portions of two real contracts. Our approach models contracts such that their robustness is assessed in a structured manner. However, many open questions remain.

First, it would be interesting to map the notions of robustness into an existing contract language, such as the one proposed by Oren et al. [15], and to automate the rules for robustness, creating an algorithm that may identify whether a contract is robust or not. It would also be useful to study a large number of additional contracts, and see whether our rules for robustness are exhaustive, or should be altered in some way. Further, it may be possible to identify additional commonly occurring classes of commitments, together with associated robustness rules.

The notion of robustness becomes increasingly important as agents autonomously negotiate and create contracts between themselves. By creating a robust contract, able to state what should occur in all situations (within the context of the contract), an agent's cognitive load is reduced, as it does not need to reason about whether the contract was adhered to or not. Further, robust contracts minimize the situations in which humans need to intervene in order to handle agent disagreements. While many open problems remain, this paper provides an initial approach to identifying and creating robust contracts.

References

1. Abrahams, A.S., Bacon, J.M.: A software implementation of Kimbrough's disquotation theory for representing and enforcing electronic commerce contracts. Group Decision and Negotiation 11(6), 487–524 (2002)
2. Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E. (eds.): Multi-Agent Programming: Languages, Platforms and Applications, Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 15. Springer (2005)
3. Bordini, R.H., Hübner, J.F., Wooldridge, M.J.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons, Chichester, UK (2007)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Journal of Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (May 2004)
5. Carbogim, D., Robertson, D.: Contract-based negotiation via argumentation (a preliminary report). In: Proceedings of the Workshop on Multi-Agent Systems in Logic Programming: Theory, Application, and Issues (MAS) held at the International Conference on Logic Programming (ICLP). Las Cruces, New Mexico (1999)
6. Daskalopulu, A.: Logic-based tools for legal contract drafting: Prospects and problems. In: Proceedings of the First Logic Symposium. pp. 213–222. University of Cyprus Press (1997)
7. Daskalopulu, A., Dimitrakos, T., Maibaum, T.: Evidence-based electronic contract performance monitoring. Group Decision and Negotiation 11(6), 469–485 (2002)
8. Desai, N., Narendra, N.C., Singh, M.P.: Checking correctness of business contracts via commitments. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS). pp. 787–794. IFAAMAS, Columbia, SC (May 2008)
9. Gisler, M., Stanoevska-Slabeva, K., Greunz, M.: Legal aspects of electronic contracts. In: Proceedings of the CAiSE Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing (IDSO). CEUR Workshop Proceedings, vol. 30. CEUR-WS.org, Stockholm (2000)
10. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems 14(2–3), 181–216 (2005)
11. Grosof, B., Poon, T.C.: SweetDeal: Representing agent contracts with exceptions using semantic web rules, ontologies, and process descriptions. International Journal of Electronic Commerce 8(4), 61–98 (2004)
12. Mallya, A.U., Singh, M.P.: Incorporating commitment protocols into Tropos. In: Müller, J.P., Zambonelli, F. (eds.) Proceedings of the 6th International Workshop on Agent Oriented Software Engineering (AOSE 2005). LNCS, vol. 3950, pp. 69–80. Springer, Berlin (2006)
13. Meneguzzi, F., Miles, S., Holt, C., Luck, M., Oren, N., Faci, N., Kollingbaum, M.: Electronic contracting in aircraft aftercare: A case study. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems. pp. 63–70 (2008)
14. Nunes, I., Cirillo, E., de Lucena, C.J.P., Sudeikat, J., Hahn, C., Gomez-Sanz, J.J.: A survey on the implementation of agent oriented specifications. In: Agent-Oriented Software Engineering X: State of the Art Survey, Lecture Notes in Computer Science, vol. 6038, pp. 157–167. Springer (2010)

15. Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a formalisation of electronic contracting environments. In: Proceedings of the International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN) held at AAAI. pp. 61–68. Chicago (2008)
16. Rodríguez-Aguilar, J.A., Martín, F.J., Noriega, P., Garcia, P., Sierra, C.: Towards a test-bed for trading agents in electronic auction markets. AI Communications 11(1), 5–19 (1998)
17. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7, 97–113 (Mar 1999)
18. Telang, P.R., Singh, M.P.: Enhancing Tropos with commitments. In: Borgida, A., Chaudhri, V.K., Giorgini, P., Yu, E.S.K. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 417–435. Springer (2009)
19. Telang, P.R., Singh, M.P.: Abstracting and applying business modeling patterns from RosettaNet. In: Proceedings of the 8th International Conference on Service-Oriented Computing (ICSOC). pp. 426–440. ACM, San Francisco (2010)
20. Telang, P.R., Singh, M.P.: Specifying and verifying cross-organizational business models: An agent-oriented approach. IEEE Transactions on Services Computing 4 (2011), in press
21. Weyns, D., Haesevoets, R., Helleboogh, A.: The MACODO organization model for context-driven dynamic agent organizations. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 5(4), 16:1–16:29 (Nov 2010)